

**McMaster University**

**Faculty of Engineering**

**Department of Computing and Software**

**COMP SCI 4ZP6**

Market Identifier

By

Ryan Ticlo (ticlor)

Kenny Kim (kimm54)

Rickesh Mistry (mistrrp)

Muyideen Jimoh (jimohma)

Supervisor

Dr. Archer

Course Coordinator

Dr. Anand

Contents

[Project Abstract 3](#_Toc7100968)

[1. Introduction 3](#_Toc7100969)

[2 Problem Formulation 3](#_Toc7100970)

[2.1 Problem Statement 3](#_Toc7100971)

[2.2 Problem Solution 3](#_Toc7100972)

[3. Specifications 4](#_Toc7100973)

[3.1 Product Functions 4](#_Toc7100974)

[4. Requirements 5](#_Toc7100975)

[4.1 Server-Side Requirements 5](#_Toc7100976)

[5. System Diagram 5](#_Toc7100977)

[6. Value Proposition 6](#_Toc7100978)

[7. Cost Analysis 6](#_Toc7100979)

[8. Significance of data 6](#_Toc7100980)

[9. Optimization of Software Implementation 6](#_Toc7100981)

[Quicksort Algorithm and Search Algorithm 7](#_Toc7100982)

[Socket Programming (Client to Server) 7](#_Toc7100983)

[Multithreading and Multiprocessing 8](#_Toc7100984)

[10. Conclusions 9](#_Toc7100985)

[11. References 9](#_Toc7100986)

[12. Appendix 10](#_Toc7100987)

[External Interface Requirements 10](#_Toc7100988)

[API Documentation 10](#_Toc7100989)

# Project Abstract

Market Identifier is a business analytics application that helps small businesses and new entrepreneurs identify profitable niche opportunities. Therefore, allowing new entrepreneurs to strategically penetrate markets with low budgets. By helping small businesses identify these niche markets, we can provide a new stream of income to entrepreneurs, stimulate the economy, and provide a better selection of goods to consumers. Market identifier’s algorithms analyze the results of the queries by the user and output an opportunity score, based on various marketing and economic indicators.

# Introduction

One of the biggest factors determining a new business’s success is timing, which is closely related to luck. This is one of the biggest reasons people avoid starting their own businesses: the risk of failure, and uncertainty of not knowing if their idea is needed and viable. Big data can help solve this problem. When start-ups are provided with in-depth market analysis, they can better identify profitable gaps to capitalize on. In order for entrepreneurs to generate income, in the presence of large corporations, they must fill markets in which big corporations are not selling in. These markets are not big enough and therefore not as desirable for larger corporations. However, there may be sufficient demand within these niche markets for individual entrepreneurs to make a sizable income off of. By helping small businesses identify these niche markets, we can provide a new stream of income to entrepreneurs and a better selection of goods to consumers.

# Problem Formulation

## Problem Statement

Uncertainty and risk associated with starting a new business venture.

## 2.2 Problem Solution

Providing business analytics to small businesses and entrepreneurs which will reduce risk and uncertainty. The application will analyze sales, rating, brand dominance, and review count to help gauge the success probability of a market.

# 3. Specifications

## 3.1 Product Functions

|  |  |
| --- | --- |
| Package | Description |
| APIScript | Contains all functions required to query data from external servers. Queries are made using API’s from Keepa (Amazon), eBay, and Walmart.  Main functions include:   * queryEbayData * queryAmazonData * queryWalmartData |
| CriteriaFilter | Contains functions which can be used to reduce the number of rows in the data set by applying upper and lower bound filters.  This functionality has been removed from the final solution because of conflicts with the MI Score functionality which is considered more valuable to the users.  Main functions include:   * FilterData * quickSort * binarySearchLow * binarySearchHigh |
| ModifyData | Contains all functions required to transform raw data from each e-commerce platform into a standard form for processing.  Main functions include:   * reduceEbay * reduceAmazon * reduceWalmart |
| MIScore | Contains all functions used to calculate the MI Score of each data set based on brand dominance, rating, and reviews.  Main functions include:   * calcScore * addBrand |

# 4. Requirements

## 4.1 Server-Side Requirements

API’s are provided by Amazon, eBay, and Walmart to retrieve data for products. Queries will be initiated from Market identifier’s AWS server to servers from the companies mentioned above, via their APIs. Quicksort and binary search algorithms are implemented for sorting and searching through the results from the queries. The result from a sort procedure will display top results using a rank field returned by Amazon. The price, sales, and feedback are not required for the algorithm but will be displayed to benefit the user.

# 5. System Diagram

The code is hosted on an AWS plan that consists of an Ubuntu server. A user interacts with the web application using their local PC, our server makes the relevant queries to retrieve data from Amazon, eBay and Walmart servers, sorts and analyzes this data, then displays it to the user in a meaningful way.

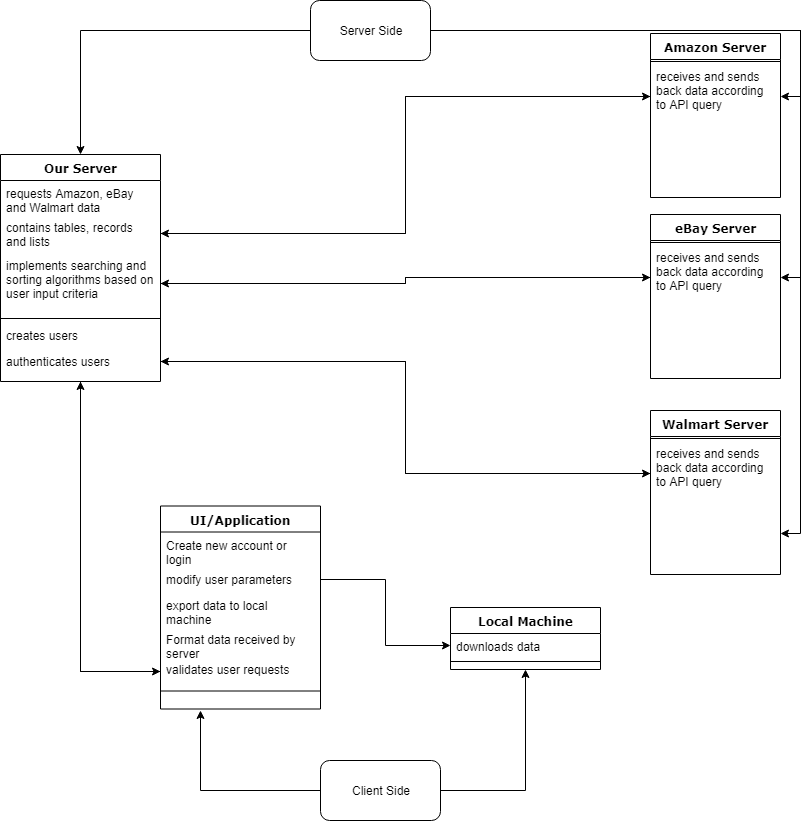


Figure (a) System Flowchart

Server location: ec2-3-91-79-241.compute-1.amazonaws.com

# 6. Value Proposition

Market Identifier provides the user with data from 3 different e-commerce platforms. Whereas, similar solutions such as Helium10, focus just on 1 platform (Amazon). Our solution also provides the user with an MI score: a score out of 4. A product that has an MI score of 1 is a good opportunity to capitalize on. While an MI score of 4 is bad. The score is determined by analyzing various key indicators, such as review count, rating, brand dominance, and price.

# 7. Cost Analysis

**API access:** $99/month

**AWS server cost:** $12.99/month

**Domain name cost:** $10/year

# 8. Significance of data

**Review count:** If there are too many reviews, it is indicative of a well-established, saturated market, with little room for differentiation.

**Brand dominance:** If there is one brand with the majority of sales, it suggests that there is brand loyalty, and consumers will be reluctant to purchase elsewhere. Thus, making it a harder niche to penetrate.

**Price:** For start-up companies in the e-commerce space, it is difficult to profit off an item that is below $10, given the increase in marketing costs.

# Optimization of Software Implementation

Our project involves the use of data information obtained from public sources, of which these data are organized in two particular data structures; dictionaries and list, and are by no means ordered in any specific sequence. As no information about the order of data wasn’t provided by our data source providers, our group made a presupposition that the obtained data were in random order, to begin with. For data processing purposes, we aimed to make our data easy to work with and manipulate by applying concepts learned from various courses taken here at McMaster University. Concepts learned from 2nd-year courses through the 4th year were introduced in order to accomplish a fast and optimized implementation of our software program. These techniques include:

1. Sort Algorithm (Comp Sci 2C03)
2. Search Algorithm (Comp Sci 2C03)
3. Socket Programming (Comp Sci 4C03)
4. Multithreading/Multiprocessing (Comp Sci 4F03)

## Quicksort Algorithm and Search Algorithm

Since our data is presumed to be non-ordered, we applied the quicksort algorithm to order it before any processing is done on the data. Sorting the data first made the process of searching items much faster based on the search algorithm implemented. Initially, a linear search algorithm was implemented, however, the runtime of the search algorithm improved from O (N) to O (Log N) based on the binary search algorithm implemented. This improvement is accomplished because the binary search algorithm required a sorted/order data which was obtained by the initial application of the quicksort algorithm.

## Socket Programming (Client to Server)

Our project involved the implementation of a back-end (i.e. server) and a front-end (for clients). This dynamic was accomplished by applying the knowledge of P2P programming learned from Comp Sci 4C03. The first peer, in this case, is the client and the second peer the server. Based on the nature of this client-server interaction, the server is always listening and ready to take a request from the client. A TCP connection is applied here to ensure security during the interaction between the clients (User) and sever (Back-end). Since our software will likely be utilized by multiple people at once, the TCP connection was a suitable protocol to use because it allowed the server side to talk to multiple clients at the same time. Every time a client accesses our website, it connects to the server by creating a TCP socket accompanied with an IP address and a source port number. Since there are multiple clients likely accessing our server at the same time, the source port numbers are used for distinguishing between the individual clients.

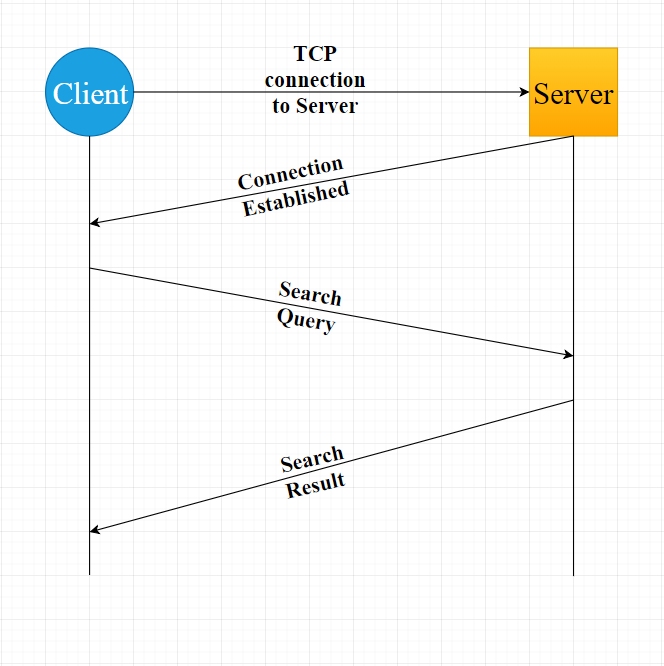


Figure (B) Client Server Flowchart

## Multithreading and Multiprocessing

The data used for this project were obtained from three different e-commerce website e.g. Amazon, Walmart, and eBay. The initial implementation of our software was initially executed in a sequential fashion. However, multithreading was introduced to split the execution into different chunks for faster execution. Since we had three websites to make API calls to, we optimized our program to fork three additional threads whereby each thread was responsible to make an API call to a respective website i.e. Walmart, Amazon, and eBay. Since these API calls are independent of each other and use no shared memory location, the threads were executed in parallel which reduced the timing by a large portion compared to the sequential execution implemented initially. Once the individual threads were done with their respective jobs, they all return and terminate (i.e. join together) to form a single thread execution. A graphical representation of the forking and joining of threads is shown below. Also when searching items in each e-commerce, we used multi-processing to achieve better usability with faster execution. We have tested out generating various number of processes to determine the most efficient number of processes due to the bottleneck (generating processes and locking the shared data). Locking mechanism was used for appending returned data to list of data.

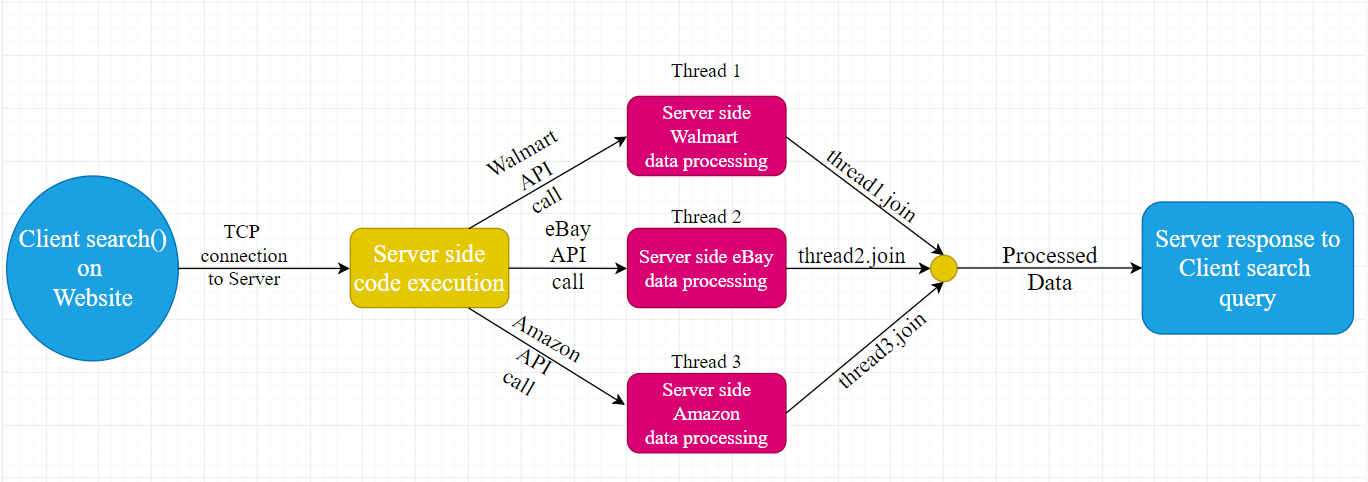


Figure (C) Forking and Joining threads

# 10. Conclusions

The final product is capable of retrieving data from Amazon, Walmart, and eBay via their respective APIs. The algorithms are capable of analyzing the results and producing an opportunity score for a particular market based on various indicators such as sales, reviews, rating, and brand dominance.

# 11. References

1. All API documentation. [Online]. Available: https://developer.ebay.com/docs. [Accessed: 18-Apr-2019]

2. “Using eBay RESTful APIs,” eBay Developers Program. [Online]. Available: https://developer.ebay.com/api-docs/static/ebay-rest-landing.html. [Accessed: 18-Apr-2019]

3. “keepa Documentation¶,” keepa Documentation - keepa 0.15.0 documentation. [Online]. Available: https://keepaapi.readthedocs.io/en/latest/. [Accessed: 18-Apr-2019].

4. “Product Lookup API,” Walmart Open API - Product Lookup API. [Online]. Available: https://developer.walmartlabs.com/docs. [Accessed: 18-Apr-2019].

5. Amazon Web Services. ec2-3-90-176-69.compute-1.amazonaws.com

# 12. Appendix

### External Interface Requirements

#### User Interfaces

API’s are provided by Amazon, eBay, and Walmart to retrieve data for products. OpenWeatherMap provides an API which will be used by our solution to retrieve current weather data. Queries will be processed using Amazon and Walmart API’s when users search for keywords or products. We will use a sorting algorithm to display top results using a rank field returned by Amazon. The price, sales, and feedback is not required for the algorithm but will be displayed to benefit the user. Weather and average income filters can be selected by users. Historical data will be searched for dates with the selected values and new queries to Amazon will be process to find sales data.

### API Documentation

##### EBay Data

Overview

eBay has several API packages available for accessing item data. With the item\_summary API, we can query on an item name and retrieve a list of item identification codes with some additional details such as price. Item id can be used with the Item API for additional information. RESTful calls using eBay APIs require authorization to be set in the request header. The required ID changes according to the API. See the official API for more details on which ID to use.

App ID (Client ID)

RickeshM-MarketId-PRD-4393cab7d-e5a6ffdf

Dev ID

30f859b7-87ae-42ca-830b-8c1c31b8e6da

Cert ID (Client Secret)

PRD-393cab7d81f4-9329-4a25-9cda-1a4f

Base64-encoded Credentials

Umlja2VzaE0tTWFya2V0SWQtUFJELTQzOTNjYWI3ZC1lNWE2ZmZkZjpQUkQtMzkzY2FiN2Q4MWY0LTkzMjktNGEyNS05Y2RhLTFhNGY=

Official API Documentation

All API Docs:

https://developer.ebay.com/docs

Useful getting started documentation for making calls:

https://developer.ebay.com/api-docs/static/ebay-rest-landing.html

Item\_Summary API documentation used for getting item listings:

https://developer.ebay.com/api-docs/buy/browse/resources/item\_summary/methods/search

Item API documentation used for specific item data:

https://developer.ebay.com/api-docs/buy/browse/resources/item/methods/getItem

Client Credentials Grant Flow API required for Item\_Summary:

https://developer.ebay.com/api-docs/static/oauth-client-credentials-grant.html

Queries

The following query was made in javascript. The headers required as specified in the official documentation have been set. A POST call is made to create an access token with the client credentials grant flow. The access token is then used as the authorization for the item\_summary call.



Response

The JSON response can be parsed to retrieve the required information for our solution.



#### Dataset

|  |  |
| --- | --- |
| Field Name | Description |
| title | The title of the product. |
| epid | An EPID is the eBay product identifier of a product from the eBay product catalog. This indicates the product in which the item belongs.  Occurrence: Conditional |
| categories | This container returns the primary category ID of the item (as well as the secondary category if the item was listed in two categories).  Occurrence: Always |
| marketingPrice | This container is returned if the item is eligible for a seller discount and contains the item's original price, and the seller discount amount and percentage.  Occurrence: Conditional |
| shortDescription | This text string is derived from the item condition and the item aspects (such as size, color, capacity, model, brand, etc.). Sometimes the title doesn't give enough information but the description is too big. Surfacing the shortDescription can often provide buyers with the additional information that could help them make a buying decision. |
| estimatedAvailabilityQuantity | The estimated number of this item that are available for purchase. Because the quantity of an item can change several times within a second, it is impossible to return the exact quantity. So instead of returning quantity, the estimated availability of the item is returned.  Occurrence: Conditional |
| primaryProductReviewRating | The container that returns the product rating details, such as review count, rating histogram, and average rating.  Occurrence: Conditional |

##### Walmart Data

Overview

Walmart Open API specifications provides several API tools including the "Trending API", "Reviews API", "Data Feed API" and "Product Lookup API". These tools require an active account with an API Key in order to be used freely by any developer. The API provides special feeds of items that are hot selling on the Walmart website. These items range from rollback and bestsellers category. Their current feeds are updated every 24 hours. The lookup API provided allows viewing of important parts of a product page to the end customers, along with the pricing status in real time. Trending API gives information on what is bestselling on Walmart.com right now. The items are curated on the basis of user browse activity and sales activity, and updated multiple times a day. A total of 5 calls can be made per second while a total of 5,000 calls can be made per day using the production API key assigned.

Dev Application ID

mktIdentif\_

Dev API Key

qufjxk28t4su7mr8sgxhpd7n

Official API Documentation

All API Docs:

https://developer.walmartlabs.com/docs

I/O Docs: Interactive API Tool for Making Calls:

https://developer.walmartlabs.com/io-docs

Product Lookup API documentation being used in our solution:

https://developer.walmartlabs.com/docs/read/Home

Queries

The following query was made for an item with id “12417832” and format of response is in json! The only required query parameter needed to make a successful query is an apiKey which is the API access key of any developer.

Sample Lookup for item id 12417832

http://api.walmartlabs.com/v1/items/12417832?apiKey={qufjxk28t4su7mr8sgxhpd7n }&lsPublisherId={Your LinkShare Publisher Id}&format=json

**Response**



Dataset

Walmart API useful fields

Reviews API

Name (Name of product (string type)

Reviews (Actual reviews submitted by people with upvotes and downvotes (string and integer type))

availableonline (Online availability of product (Boolean type)

salePrice (Market price of product (integer type)

reviewStatistics (ratingValue (i.e 1,2,3,4,5) and number of counts such a rating has)

Search API

totalResults (number of search results related to item searched for)

numItems (This is the actual number of items returned that matches the search field)

Items API

numReviews

Stock (Availablity of item on the website. Value is of type boolean (True or False))

salePrice (Market selling price of product on website (integer type))

customerRating (This gives the overall rating of item based on the ratings posted by customers (Rating scale 1 to 5))

##### Amazon Data

Overview

Keepa is an API tool that enables users to request items on Amazon and returns the result. It can be sorted by review count, price, rating etc. The request can be written in Java and Python.

Interfacing with the Keepa requires a valid access key. This requires a monthly subscription from Pricing. Here’s a brief description of the subscription model from their website: “All plans are prepaid for 1 month with a subscription model. A subscription can be canceled at any time. Multiple plans can be active on the same account and an upgrade is possible at any time, a downgrade once per month. The plans differentiate by the number of tokens generated per minute. For example: With a single token you can retrieve the complete data set for one product. Unused tokens expire after one hour. You can find more information on how our plans work in our documentation.”

Keepa access key ID:

AKIAIJPGALEGINWEHUDQ

Getting started: Making your first request:

**Connecting to KeepaAPI:**

import keepa accesskey = 'XXXXXXXXXXXXXXXX' # enter real access key here

api = keepa.Api(accesskey)

**Making your first request:**

<https://buildmedia.readthedocs.org/media/pdf/keepaapi/stable/keepaapi.pdf>

**Components of a request:**

<https://keepa.com/#!discuss/t/how-to-make-requests/767>

<https://keepa.com/#!discuss/c/api/apirequests>

ItemSearch

•SalesRank (Rank of item ranked along items in the search pool (integer type))

•MaximumPrice/MinimumPrice (field that will can be used to return items in the price range (xml/json type))

•Availability (description of availability of items in the search pool (xml/json type))

•SearchInside (from the search pool, returns search results with additional parameters (xml/json type)

•VariationSummary( provides lowest price, highest price, lowest sale price, and highest sale price for all child item in search pool (xml type))

•SearchBins(Narrowing search results by Brand Name, Price Range, Subject, Percentage Off)

Response Format:

-snippet of the response from Keepa api

