Project Requirements Document v1

Company: CJ Affiliate

Project title: The Great, Fantastic Project

Team: The Great, Fantastic Team

Team Lead: Kyle Ng

Team Scribe: Nathan Guan

Team Members and Emails:

- Joey Zhang (chongyu@ucsb.edu)
- Nathan Guan (nguan@ucsb.edu)
- Kyle Ng (kyleng@ucsb.edu)
- Minliao Li (minliao@ucsb.edu)
- Zhuo Chen (zhuo@ucsb.edu)

Background:

Today's affiliate business sees a world blooming with opportunities, millions of advertisement offers lack the means to connect with a provider while the providers struggle for a perfect fit. The providers don't have to be professional enterprises. Take the example of home bloggers. Home bloggers blog about something they are passionate about and they want to share their passion with their readers. For them to recommend products, they would need to conduct hours and hours of research. Not only that, new products are added to the market every day and prices are always in flux. There is currently no way for the blogger to do all this research in a timely manner and still provide the best offers/products to their loyal readers.

Goal:

The goal of this project is to provide a way for bloggers to recommend deals and/or projects to their readers based on their blog topic(s). With this product, they input keywords from their blog that they want offers for. From there, our web application will supply offers based on those keywords to the blogger in the form of a list. The blogger will be able to select which offers they want to post onto their site through a HTML snippet.

Input:

The input of the web application will be selected from a list of keywords. Once selected, the algorithm will take into account the current trends from an external source (Twitter for example) and produce an improved input using them, it will create a query for our backend server.

Output:

The output on the web application will be a list of offers best suited for the user based on their selected keywords. This output will most likely contain multiple offers for any combination of keywords selected. From the offers displayed to the user, one will be able to select however many of the offers that they want to use for their blog. The process for getting the offers a user wants onto their blog will be easy because they will be able to get an HTML snippet from our website that can be pasted onto their blog. This HTML snippet will then display the relevant offer on their blog once embedded in said blog.

Procedure:

We will produce an algorithm that will take in sample data from CJ and filter out the irrelevant products and offers. Then we'll pass these filtered results into another database which is ready for presenting to the user. Then we will display all of the offers fitting the user's need (the user's need is defined by their inputted or selected keywords at this stage) on a separate web application. We will also build an algorithm which promotes ads based on our filter (in this case, our filter is the trending on social media). This will allow bloggers to advertise ads that's interesting to the viewers, maximizing the profit they may achieve through their blogs.

Milestones:

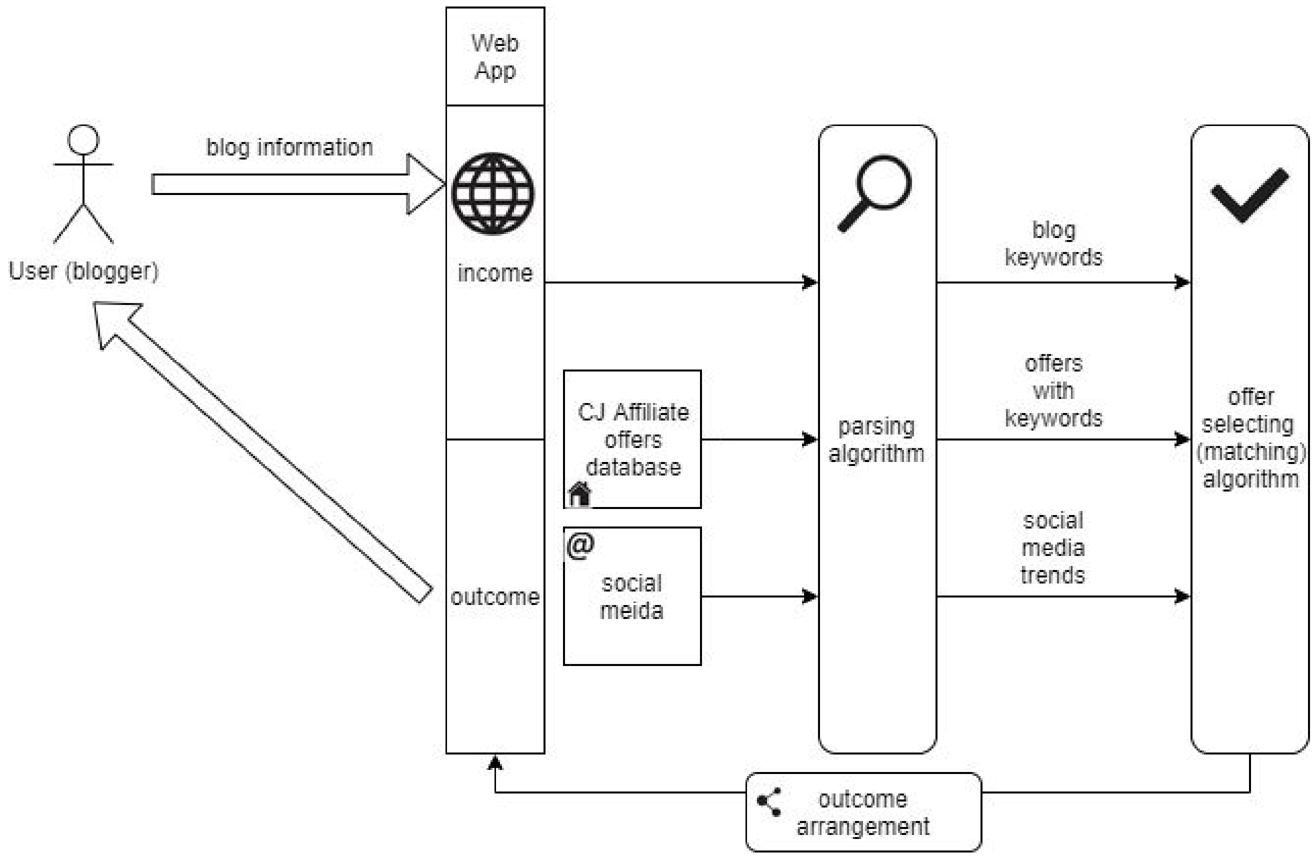
- MVP: Web app with a display of "good" offers
- Algorithm that uses keywords as input and output data with those key works in the description
- Takes in sample data instead of hard-coded data

- Search bar for list of keywords
- Selectable keywords for blogger to use
- Buttons to start the sorting process
- Output HTML/JavaScript code for the blogger to post on their blog

Technologies we plan to use:

- Heroku
- Code from CJ
- Frontend: HTML, CSS, JavaScript, ReactJS
- Backend: Java, Python
- Natural Language Process API

Diagram:



Use Cases:

1. **As the web application, I should be able to have a form to input text and display the offers in a table.**
   Test: Have an input box that can take in text and output table that can display offers
   https://github.com/minliaoli/CJ-Affiliate-Project/commit/a837e034a84b4e96cd6142eac0590dcc5fcb7b2f

2. **As a blogger, I want to click one button to select/deselect all keywords, so I can choose from keywords more quickly.**
   Test: Select all keywords by clicking the select button and deselect all by clicking the deselect button.
   https://github.com/minliaoli/CJ-Affiliate-Project/commit/0627151c678a81a33cfef6c861dfaf67715382e6

3. **As a blogger, I want to be able to sort the offers presented to me on the web by features, so I can choose the offer that I think is best.**
   Test: I select a sorting criteria, The offer displayed on the web changes in correspondence.

4. **As a blogger, I want to get the contact information from the app so I can contact CJ Affiliate.**
   Test: There are two buttons on the main page which show contact and about, I click on both and get two pages that's related to this information.
   https://github.com/minliaoli/CJ-Affiliate-Project/commit/9e92e20871e61308c84428786bc428ec38de9908

5. **As a parsing algorithm, I can query offer data from Firebase so I can populate the server with data.**
   Test: I ask the firebase for a piece of unparsed offer, I get this offer and output a parsed result that's ready to be put into another database.
   https://github.com/minliaoli/CJ-Affiliate-Project/commit/21f6a33412dc079516577f5f24c7fd86e5d4c221

6. **As a parsing algorithm, I can look at current trends on Twitter to output more relevant offers to the user.**
   Test: The algorithm runs a GetTrend method whenever a timer is called, then the GetTrend method returns a list of current trending keywords to the algorithm.

7. **As a blog parser, I can take in a block of text and extract and output keywords from it. Then I can classify these keywords into broader categories which can be used to match specific categories from offers.**
   Test: I can take a segment of text as input, I can output a list of keywords that's related to the text, as well as the tag that's linked to these keywords.
   https://github.com/minliaoli/CJ-Affiliate-Project/commit/75e105827caecf1968f16a8bf76b1ac85dbc9e7d

8. **As a blogger, I want to get the HTML/JS code for the selected ad so that I can paste it onto my website to display the ad.**
   Test: The HTML/JS code for all possible ads on the website is able to be copied by the user.

9. **As the offer selection algorithm, I want to be able to output the top 5 offers based on trends and keywords inputs.**
   Test: I have a list of trending keywords and parsed offers in database, I match these and assign high weight to offers that matches best with the trend. I then output a list of 5 offers which have the highest weight.

10. **As a blogger, I want to be able to login to access the web application and access the offers recommended to me in history.**
    Test: Login screen that uses Firebase authentication to see if login is accepted, Then pops the user into a personal page which contains history offers.

Overview:

- Sprint 1:
  - Choose tools or APIs to use
  - Outline workflow
  - Work on front end web app and tool to read in offers (seperate for now)
- Sprint 2:
  - Continue working getting web app up and running
  - Work on simple algorithm that takes in keyword and output data
  - Combine tools with web application front end
  - PRDv1 release

    - Work on test cases
    - Get database up and running
- Sprint 3:
    - Work on tool to sort relevancy of offers
    - Improve algorithm with the aforementioned tool
    - Improve web app with better UI
    - Develop more test cases
- Sprint 4:
    - Assure bugs are fixed and MVP is ready for short demo
    - Test cases pass
    - All hard coded data is completely removed