Project Requirements Document v2

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:

- Kyle Ng (kyleng@ucsb.edu)
- Nathan Guan (nguan@ucsb.edu)
- Minliao Li (minliao@ucsb.edu)
- Zhuo Chen (zhuo@ucsb.edu)
- Joey Zhang (chongyu@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 a block of text to be parsed in to keywords to find offers that are relevant to the blog's topic. 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. Additionally, the blogger may be able to save the offers for a future date and filter the offers.

Input:

The input of the web application will be text that the user wants to find offers for. From there, the block of text will be sent to the server so that it can parse and find relevant keywords to use to query the database. The block of text could be of any length and could contain anything that the user wishes to be included. Preferably, the input text has a topic in mind that does not digress from the main topic at hand, but the machine learning algorithm will hopefully weigh the irrelevant keywords differently. We are assuming that the blogger will provide the application with a couple paragraphs of text with some sort of topic in mind.

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:

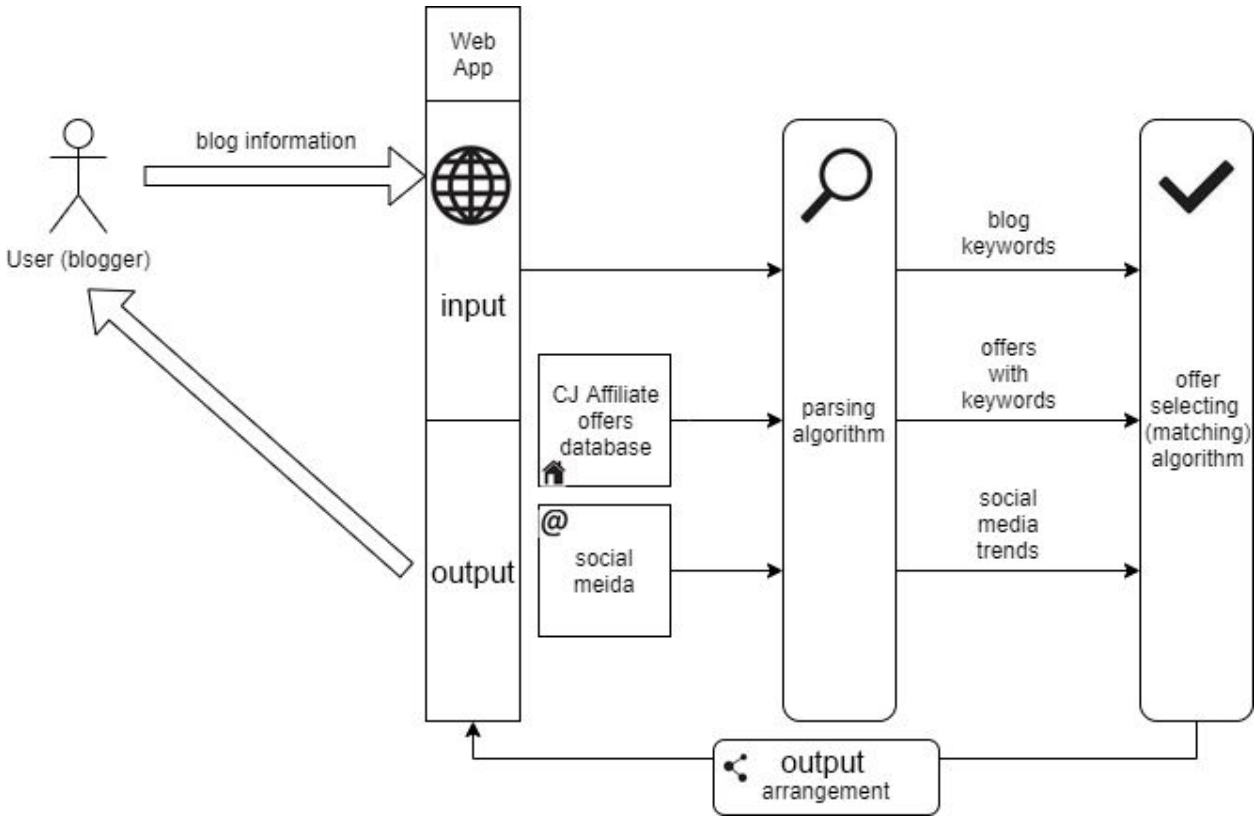
We will produce an algorithm that will take in sample data from CJ and filter out the irrelevant products and offers. Then we'll pass these filtered results into another database which is ready for presenting to the user. Then we will display all of the offers fitting the user's need (the user's need 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. We plan on using AWS Amazon Comprehend to assign weights to certain words in the text after we parse it so we can determine the correct offers to present to the user. We also plan on using Google Trends API and potentially Twitter API to find current trends to help filter our

Milestones:

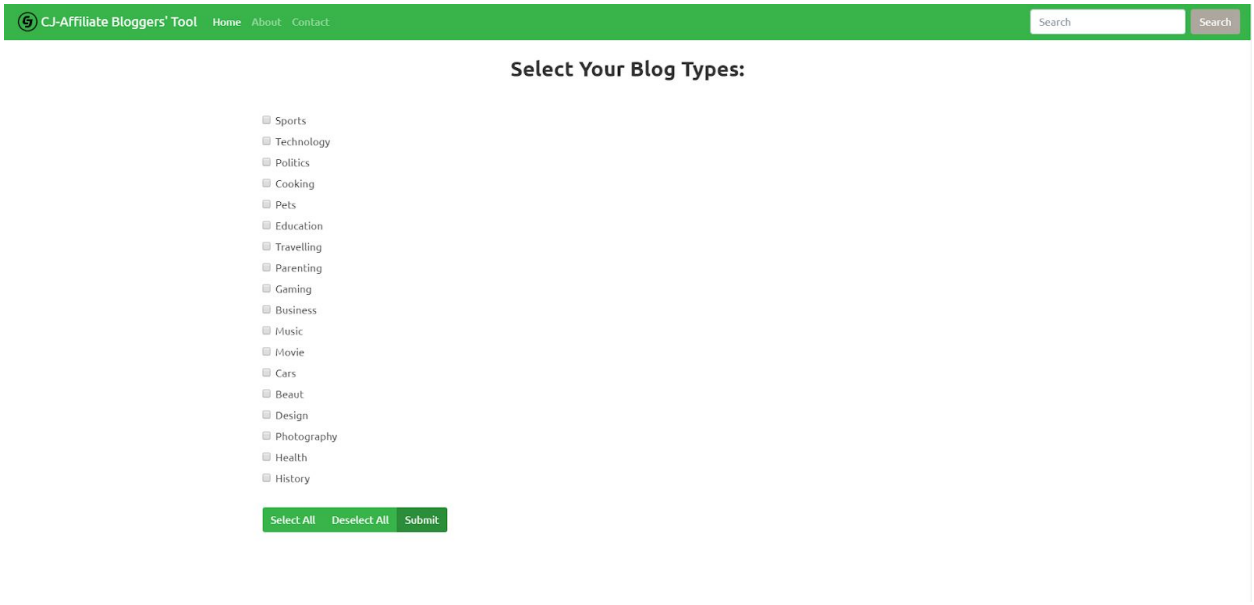
- MVP: Web app with a display of "good" offers
- Algorithm that uses keywords as input and output data with those key works in the description
- Takes in sample data instead of hard-coded data

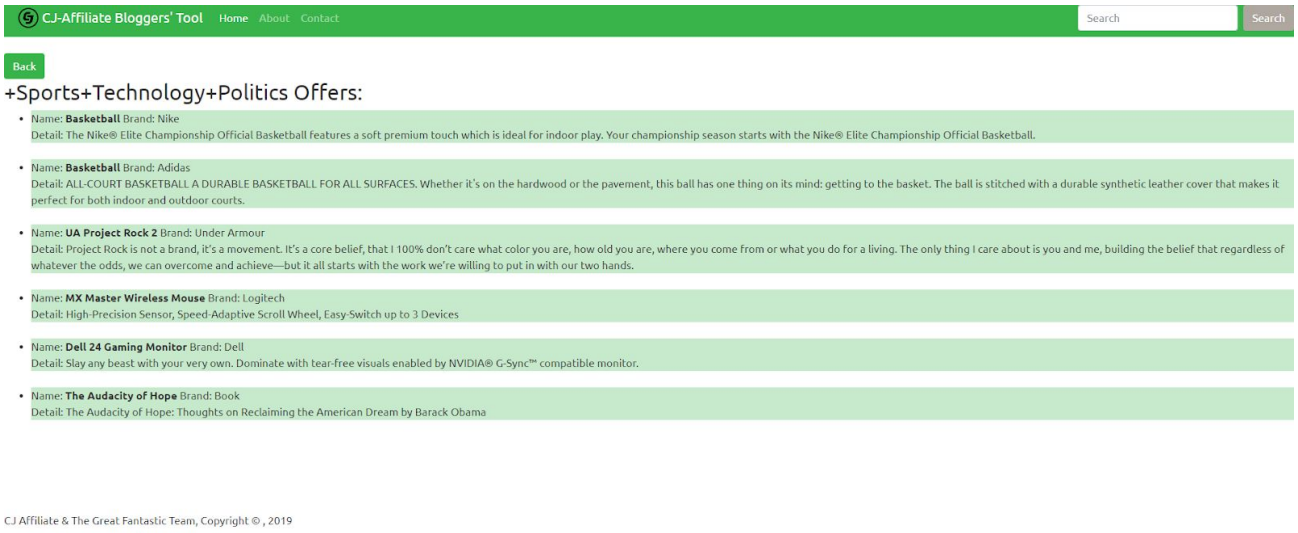
- Search bar for list of keywords
- Selectable keywords for blogger to use
- Buttons to start the sorting process
- Output HTML/JavaScript code for the blogger to post on their blog

High Level Diagrams:



UI Design:


CJ-Affiliate Bloggers' Tool
Home
About
Contact
Search
Search
Select Your Blog Types:
Sports
Technology
Politics
Cooking
Pets
Education
Travelling
Parenting
Gaming
Business
Music
Movie
Cars
Beaut
Design
Photography
Health
History
Select All
Deselect All
Submit


CJ-Affiliate Bloggers' Tool
Home
About
Contact
Search
Search
Back
+Sports+Technology+Politics Offers:
Name: Basketball Brand: Nike
Detail: The Nike® Elite Championship Official Basketball features a soft premium touch which is ideal for indoor play. Your championship season starts with the Nike® Elite Championship Official Basketball.
Name: Basketball Brand: Adidas
Detail: ALL-COURT BASKETBALL A DURABLE BASKETBALL FOR ALL SURFACES. Whether it's on the hardwood or the pavement, this ball has one thing on its mind: getting to the basket. The ball is stitched with a durable synthetic leather cover that makes it perfect for both indoor and outdoor courts.
Name: UA Project Rock 2 Brand: Under Armour
Detail: Project Rock is not a brand, it's a movement. It's a core belief, that I 100% don't care what color you are, how old you are, where you come from or what you do for a living. The only thing I care about is you and me, building the belief that regardless of whatever the odds, we can overcome and achieve—but it all starts with the work we're willing to put in with our two hands.
Name: MX Master Wireless Mouse Brand: Logitech
Detail: High-Precision Sensor, Speed-Adaptive Scroll Wheel, Easy-Switch up to 3 Devices
Name: Dell 24 Gaming Monitor Brand: Dell
Detail: Slay any beast with your very own. Dominate with tear-free visuals enabled by NVIDIA® G-Sync™ compatible monitor.
Name: The Audacity of Hope Brand: Book
Detail: The Audacity of Hope: Thoughts on Reclaiming the American Dream by Barack Obama
CJ Affiliate & The Great Fantastic Team, Copyright © , 2019

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.
   Time Estimate: 1 day
   https://github.com/minliaoli/CJ-Affiliate-Project/commit/a837e034a84b4e96cd6142eac0590dcc5fcb7b2f

2. **As a blogger, I want to see a diagram of the trends online, so I can get a sense of what is popular.**
   Test: Click on 'Get Offers' button and then a sidebar will pop up with charts and metadata.

   Time Estimate: 3 days
   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, then the offers displayed on the website changes to correspond to the selection.
   Time Estimate: 2 days
   https://github.com/minliaoli/CJ-Affiliate-Project/issues/8

4. **As a offer selection algorithm, I want to take seasonal trending (like holidays, sports season and so on) into my measurement.**
   Test: seasonal trending factor added into the algorithm.

Time Estimate: 4 days

https://github.com/minliaoli/CJ-Affiliate-Project/commit/7c26b17271f26bf6ddf1706644f11ef0575fdf0d

5. **As a parsing algorithm, I can query offer data from Firebase so I can populate the server with data.**

   Test: I can query the database with the parsed keywords and return any offers that contain those words.

   Time Estimate: 7 days

   https://github.com/minliaoli/CJ-Affiliate-Project/commit/21f6a33412dc079516577f5f24c7fd86e5d4c221

6. **As a trends reading interface, 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.

   Time Estimate: 5 days

   https://github.com/minliaoli/CJ-Affiliate-Project/blob/Google_Trends/pytrends/dailydata.py

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.

   Time Estimate: 14 days

   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.
   Time Estimate: 1 day
   https://github.com/minliaoli/CJ-Affiliate-Project/blob/loopback_react/client_src/src/components/GetOffer.js

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.
   Time Estimate: 5 days
   https://github.com/minliaoli/CJ-Affiliate-Project/blob/loopback_react/client_src/src/components/GetOffer.js

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 authentication to see if login is accepted,
    Then pops the user into a personal page which contains history offers.
    Time Estimate: 4 days
    https://github.com/minliaoli/CJ-Affiliate-Project/blob/master/OfferManager/offerManager/offers/pyrebase_settings.py#L17-L20

11. **As a blog-offer matching algorithm, I can take the keywords generated by both the blog parser and offer parser in database, match and rank them**

based on the weight suggested by the trends interpreter, then output to database.

Test: Be able to obtain the ads that correspond to current trends with the highest ranked ones first.

Time Estimate: 7 days

https://github.com/minliaoli/CJ-Affiliate-Project/blob/Google_Trends/DetectKeyPhrases

12. **As a react framework, I want to be able to make post/get requests to MongoDB, so that I can talk to the offer algorithm through a medium.**

    Test: Be able to communicate with MongoDB to get desired information about the offers.

    Time Estimate: 10 days

    https://github.com/minliaoli/CJ-Affiliate-Project/blob/python_back_react_front/OfferManager/offerManager/manage.py#L28-L32

13. **As a trend interpreter, I can take the trends provided to me by a trend reading API and generate a weight sheet for the matching algorithm, then output results to a database. This weight sheet can determine the rank of offers.**

    Test: Obtain a weight sheet with the ranks of offers based on current trends that are going to be considered.

    Time Estimate: 3 days

14. **As an offer parser, I want to be able to grab offers from the database, parse its keywords, and store them into a database.**

    Test: Be able to see all of the keywords for specific offers in a database.

Time Estimate: 7 days

https://github.com/minliaoli/CJ-Affiliate-Project/blob/python_back_react_front/OfferManager/offerManager/manage.py#L28-L32

15. **As a scheduler, I can update all the data and algorithms stored in the backend periodically. I make calls to all the functions that update these procedures every 7 days, then the database updates.**
    Test: Able to have an up-to-date database so that offers displayed on the website are always relevant.
    Time Estimate: 10 days

16. **As a blogger, I want to see the webpage with CJ aesthetic styling.**
    Test: Webpage is styled with CJ Affiliate color scheme and/or CJ frontend components.
    Time Estimate: 2 days

17. **As a react framework, I can forward the text input from a blogger to the backend. If it is too large, I can break it into fragments.**
    Test: Be able to obtain a blogger's inputted text in the backend.
    Time Estimate: 10 days

18. **As a blogger, I want to be able to choose from three different kinds of input methods: choosing keywords, inputting text, or inputting blog url.**
    Test: Have a UI on the website with the three various input types.
    Time Estimate: 2 days

19. **As a blogger, I want to see suggested offers listed on pages, and I want to be able to jump to any page and choose how many offers are listed on each page.**

Test: Have a UI that allows the user to select number of offers per page.

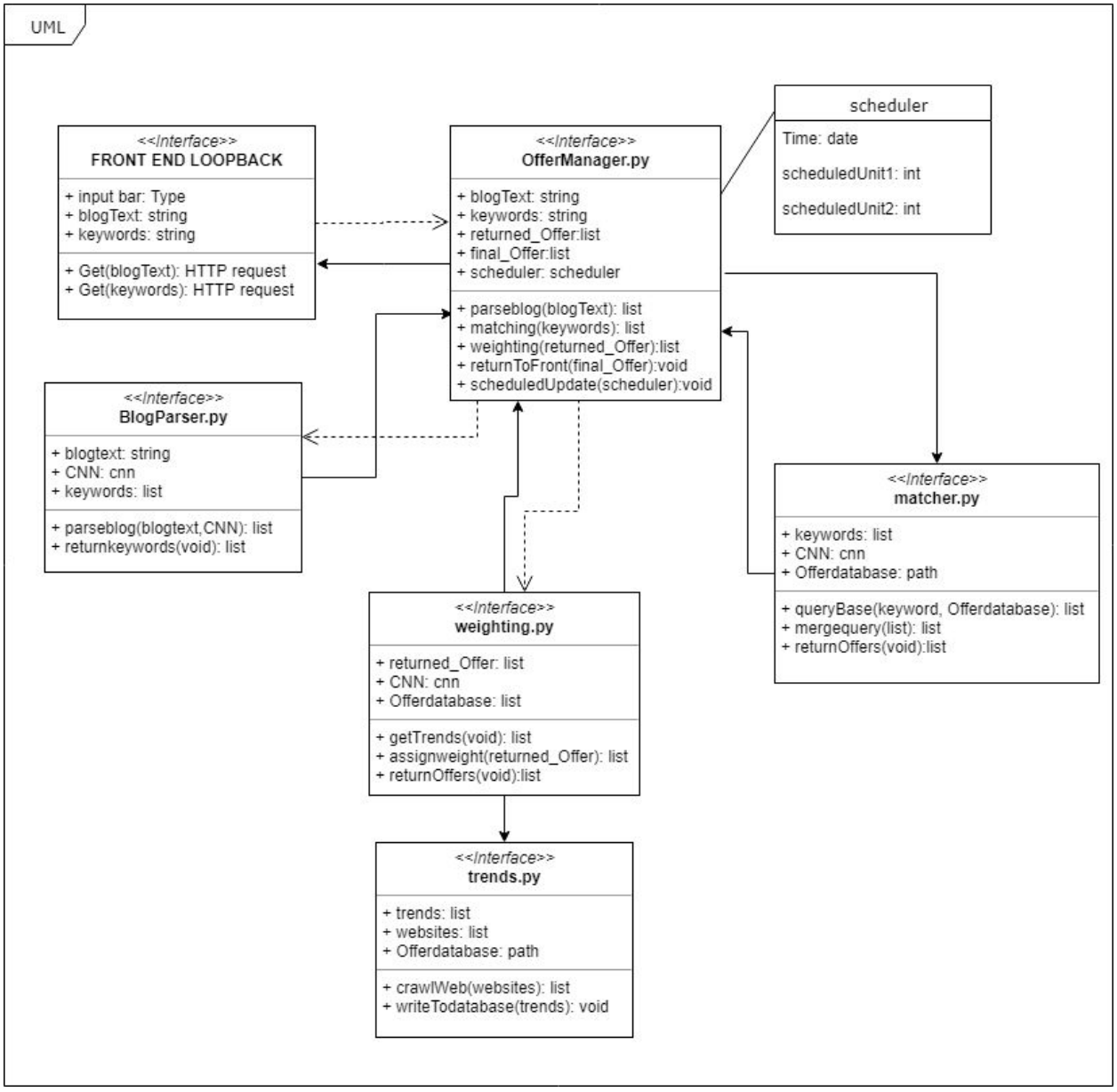
Time Estimate: 5 days

20. **As a blogger, I want to be able to know when offers expire and when new offers will be reloaded.**

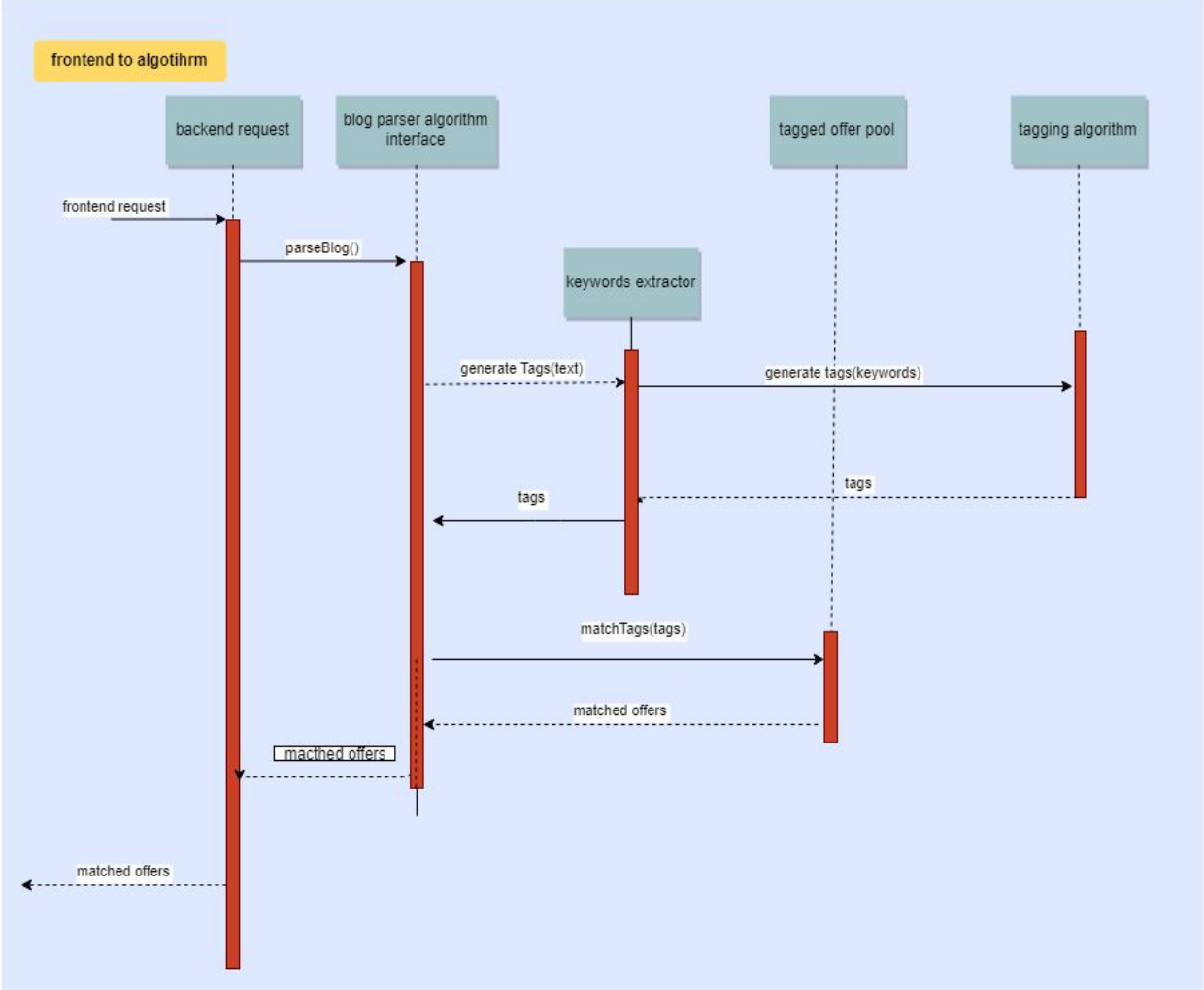
    Test: Query the server to see if offers have expiration dates and display them on the output page. Have an option to see if new data is loaded or not.

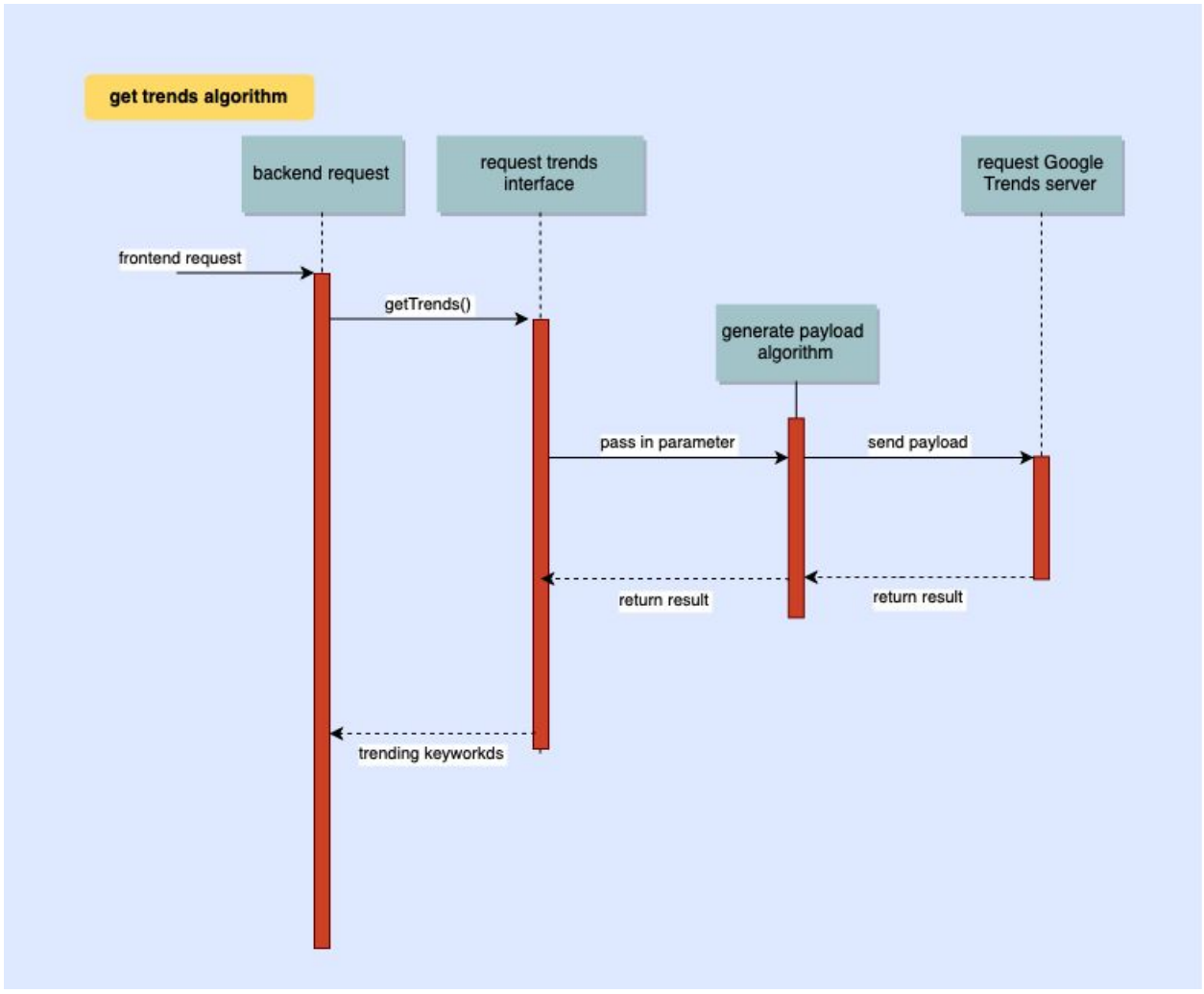
    Time Estimate: 1 day

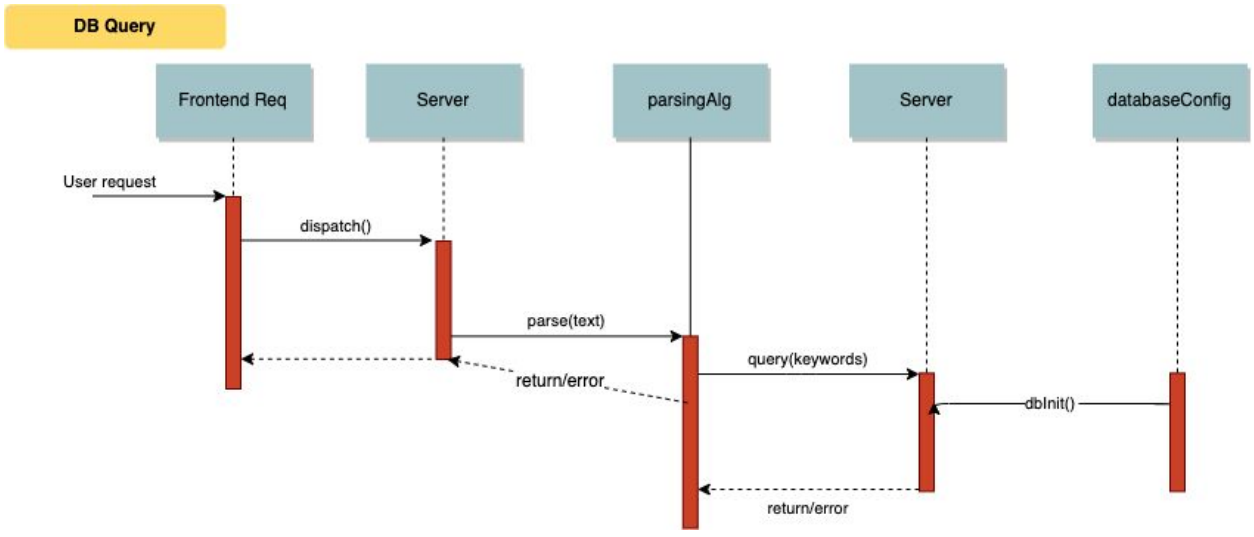
UML Diagrams:



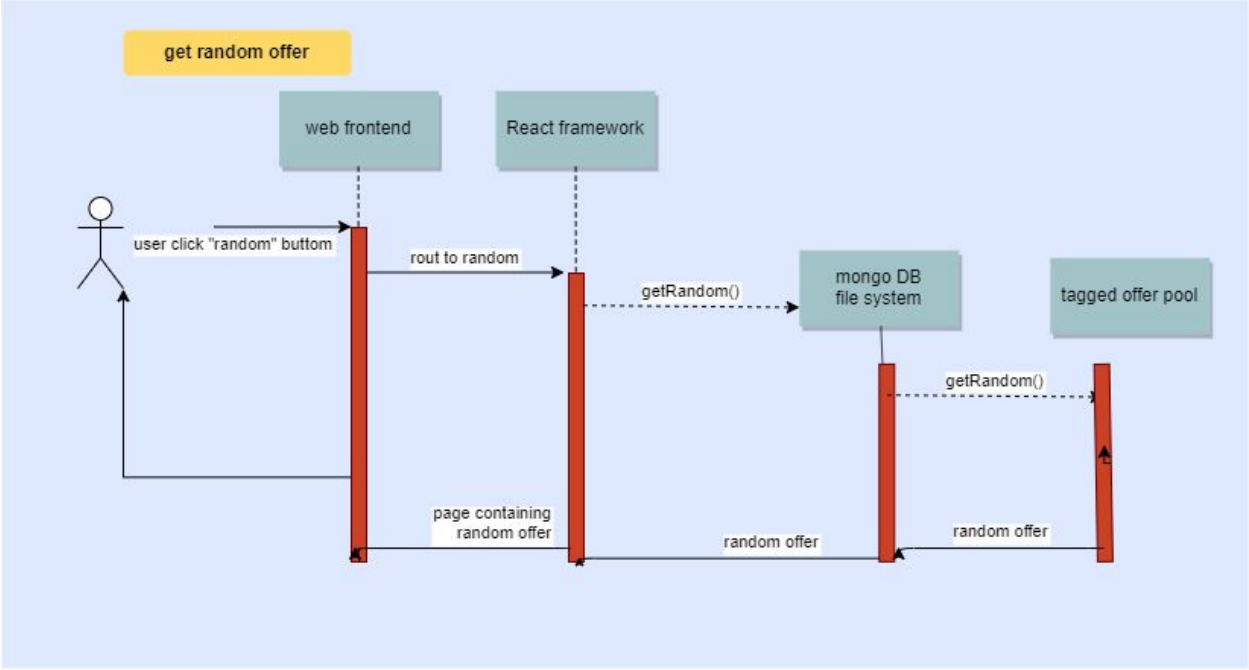
Class Interaction Sequence Diagrams:

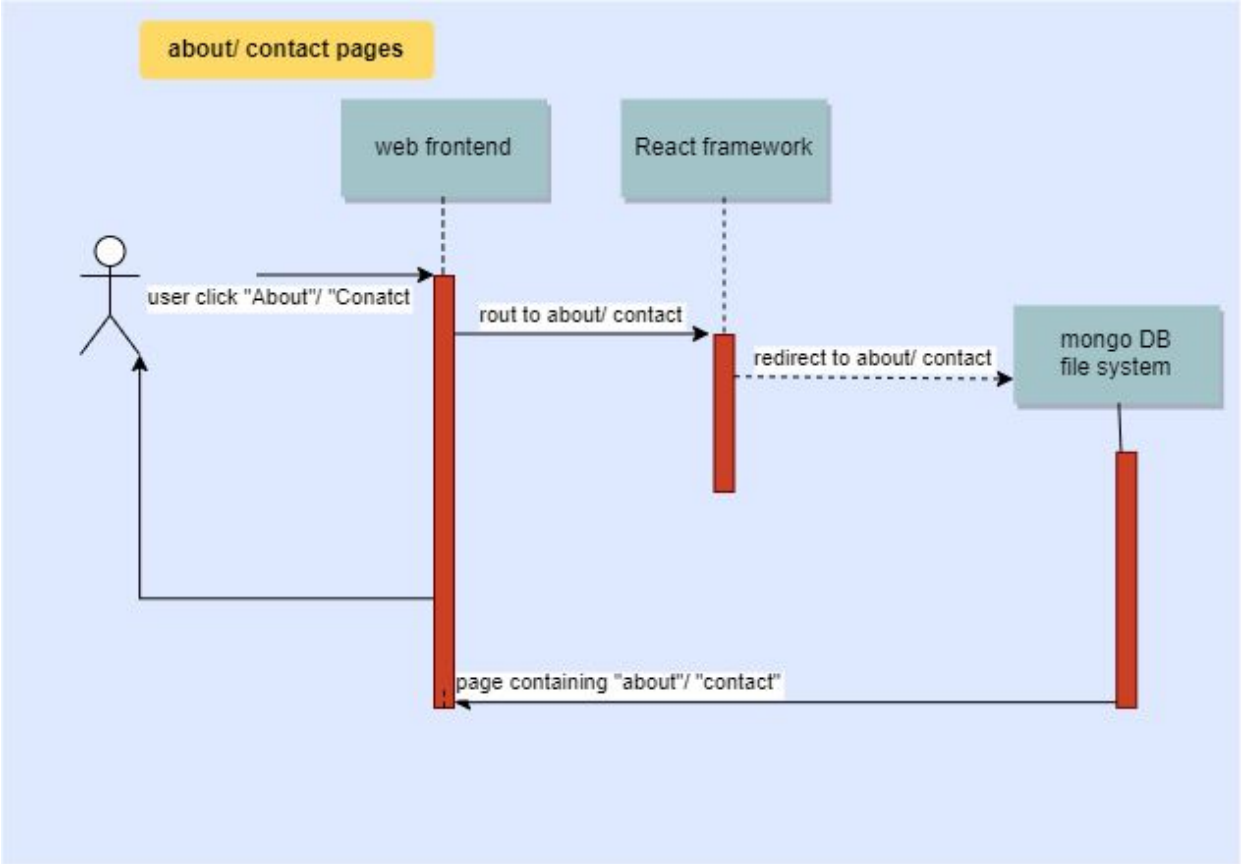


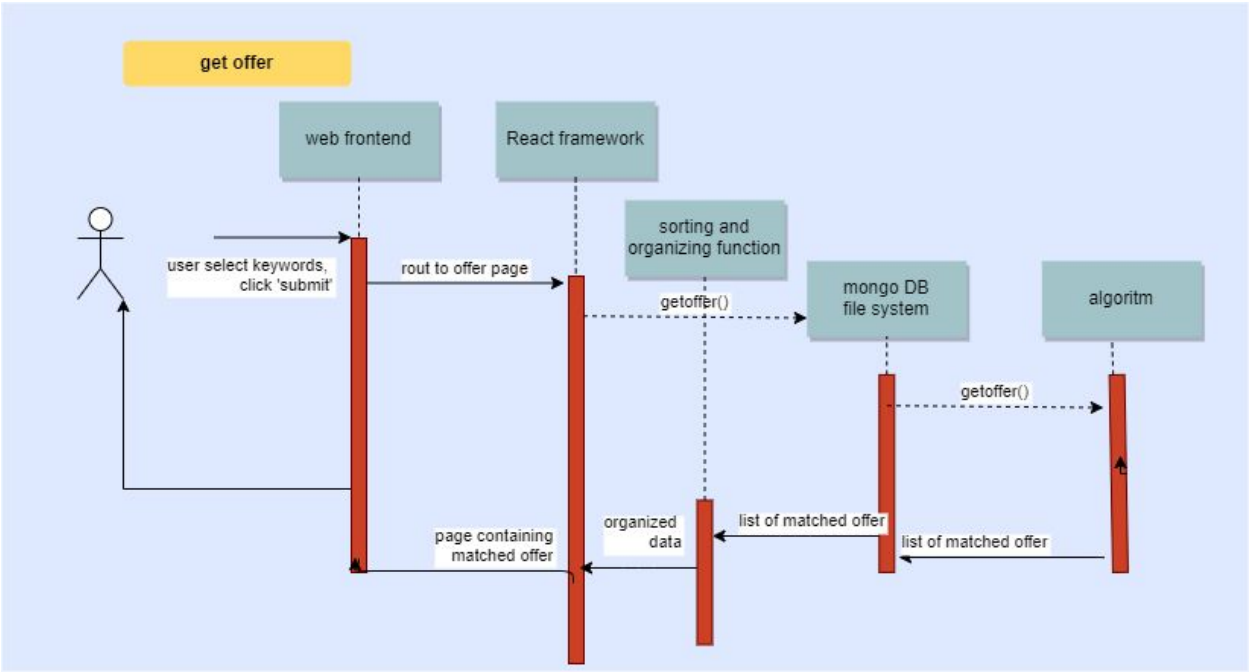

get trends algorithm
backend request
request trends interface
request Google Trends server
frontend request
getTrends()
generate payload algorithm
pass in parameter
send payload
return result
return result
trending keyworkds


DB Query
Frontend Req
Server
parsingAlg
Server
databaseConfig
User request
dispatch()
parse(text)
return/error
query(keywords)
dbInit()
return/error

User Interaction Sequence Diagrams:






get offer
web frontend
React framework
sorting and organizing function
mongo DB file system
algoritm
user select keywords, click 'submit'
rout to offer page
getoffer()
getoffer()
list of matched offer
list of matched offer
organized data
page containing matched offer

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

Appendix:

Technologies Employed:

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