Pool Party: A Vibrant, Grouped Job Board

[Presentation Link](https://www.canva.com/design/DAFUhJJlKrY/zrkgSv_ZpOFdKKNVkjHNtw/view?utm_content=DAFUhJJlKrY&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton)

# Motivation

Throughout history and especially recently, society has been growing increasingly complex, leading to an assortment of diverse occupations. It stands to reason that the job market should be full of possibility, especially for a young, well-educated applicant. In practice, as many college students entering the workforce can attest to, the job application process, which has moved primarily online, can feel fruitless and unresponsive. A sort of abundance is indeed reflected in online job boards, but the way it is implemented can often be counterproductive to facilitating efficient application, matching candidates to jobs that suit their interests and capabilities.

Leading online job boards often present an overwhelming experience, as they tend to be text-heavy with relatively little standardization. Effectively participating in these job markets is exhausting, requiring time and energy to manually sort through very individualized job postings. Important information is not clearly identifiable, instead buried in pages of text presented all at once. Users will often need to carefully parse through the content, in order to ensure that qualifications are met, then to understand and evaluate the position, all before even preparing application materials.

# Objectives

The existing online job marketplace feels crowded, tedious, and inefficient. We want to introduce a new solution, centered around large scale, at-a-glance, user-centered insight. In particular, we have 3 primary goals:

1. Characterize job postings properly, so that we can also characterize job market segments: we want to process job postings with the aim of identifying and clarifying important details.
2. Group job postings into job market segments through a variety of metrics: we want to visualize data in a way that provides insight about how the job market is segmented.
3. Allow input processing that compares user data to our job database: we want to help applicants position themselves, by allowing users to filter for their preferences and also produce data from processing user-uploaded materials.

# Data Backend

We strive for our job data to be focused and purposeful. After retrieving the data, we maintain discrete, transformed attributes to promote clarity and user-friendliness.

The initial version of our platform operates on a stagnant table of 2000 job postings from Adzuna API and The Muse API (1000 each), which are either remote or located in the United States of America, extracted on November 17, 2022. While we retrieve a sizable amount of job postings, we know it is still limited. We defer primarily to the APIs’ algorithms for relevance order to determine which 1000 postings are retrieved. We use very generalized API requests, relying on our own implementation for specific filtering.

One predetermined aspect of our API requests has to do with resolving a fundamental difference between the 2 APIs we use. A request to Adzuna API requires a country parameter, while a request to The Muse API can only specify location through cities. Luckily, [The Muse’s API documentation page](https://www.themuse.com/developers/api/v2) includes apparently all the locations you can use as a parameter when making a request to The Muse API. In this [Google Colab notebook](https://colab.research.google.com/drive/1E90-d0EFe6VUDCy1-Uc5CvL9bVo0HaUc?usp=sharing), we scrape the locations (as well as the categories, though we don’t use them in this version) from the documentation page, creating the following:

* SQL Database / TheMuseAllCats : column “Category”
* [TheMuseAllLocs](https://drive.google.com/file/d/1sT4mOCR1xa5exSZQEAEIUKg1eQibFn5l/view?usp=share_link) : columns “area” is a country or state, “city0”, “city1”, … are all the cities in the area
* SQL Database / TheMuseUSLocs : columns “City” and “State”
  + identified by assuming the format to be city, [2 capital letters for state]
* 4 URLs, which each represent a request to The Muse API with about ¼ of the US cities (~548) listed as parameters
  + After some trial and error, 548 seems to be about the maximum number of cities a single request to The Muse API can take with producing an error
  + Will be used as hardcoded request strings for API data retrieval, requesting ¼ of the total jobs requested from ¼ of the total US cities

To expand our platform to support other countries, we could use TheMuseAllLocs (defined above), after finding each country’s code and converting non-ASCII characters (which we don’t do yet and is presumably why this table could not be uploaded to SQL successfully). Then, we’d have 2 options:

1. For each country, repeat the existing process to get N jobs:
   1. Find that there are n cities in the country according to The Muse documentation
   2. Split the n cities into m relatively even groups of < 548 cities each. For each group, request m/n \* N jobs using a request string with all cities in the group
2. Get N jobs from The Muse API using no location parameters. For each country:
   1. Count that there are C returned jobs where the location is in the country
   2. Request C jobs from Adzuna using the country as the parameter

As for how we manage the content of the API data we request, we maintain a database of job data with categorical, functional attributes. In this [Google Colab notebook](https://colab.research.google.com/drive/1rUwCaekTr7JhBM_a9gJ4sru0WAL5ifGP?usp=sharing), we retrieve, transform, and store data in one streamlined process. To generalize, the notebook creates one table:

* SQL Database / jobdata
  + It is currently written to write to jobdataTEST so as to not overwrite the data that our first version operates on.
  + It is currently written to produce n=4 results from The Muse and n=4 results from Adzuna for ease of testing (getMuseJobs(4, jobs), getAdzunaJobs("us", 4, jobs)).
  + The code retrieves n job postings from each API (in the manner described above for The Muse - 4 requests, each retrieving n/4 jobs from each group of ~548)
  + The API responses are slightly different between the 2 APIs, but we match and transform the results so our database contains the following columns:
    - Most straightforward: Title (although this is somewhat inconsistent, might include location/company), Company, Posted Date, Redirect URL
    - Still slightly unbalanced between the APIs:
      * Category
        + we could potentially map Adzuna categories to The Muse categories, as The Muse categories are defined on the documentation page
      * Salary Predicted: 1 or 0
        + not very meaningful, and also not included in The Muse (so set to 0)
      * Levels: Entry Level, Mid Level, Senior Level, management, Internship
        + not included in Adzuna (so set to NA)
    - Location, requiring significant transformation
      * Location - City for The Muse; City, County or other variant for Adzuna
      * Longitude/Latitude
        + Not included in The Muse, retrieved using Mapbox API with Location
      * State - Full state name
        + Using hardcoded dictionaries from abbreviation to full name and vice versa - extracted (and mapped, for The Muse) from Location
    - Description, requiring significant transformation
      * Description
        + the most unstructured, text-based, impracticable attribute in both APIs
        + Limited to ~500 characters in Adzuna
        + Includes formatting marks (most HTML tags removed using re) in The Muse
      * Our goal is to have attributes that are categorical, instead of having to rely on reading long strings. We rewrite this description into 4 new attributes, each of which should be more focused (given a prompt) and concise (given a character limit), using OpenAI (sending the description string with a prompt concatenated, and storing the returned string in the correct column)
        + Work: What will you be doing if you work with us?
        + Hard Skills: What all is the most important background, experience, or knowledge you need to be successful?
        + Soft Skills: What are the most important soft skills you need to be successful?
        + Education: What education do you need to work in this position?

# Analytics Frontend

We generate our frontend in this [Google Colab notebook](https://colab.research.google.com/drive/1mmhINqwxjTZM0F1Dum2MohJOOdjlOaQH?usp=share_link). After cells 1-8 execute, the link to the website will be printed. There are 4 pages:

* Home
  + The home page includes a search bar that accepts a keyword search term, which will be used to filter rows with job titles that include the keyword. The home page also includes a resume upload section, which will be encoded using an nlp library, which will then be used to compare filtered rows (also encoded) to run similarity scores. Based on the filtered rows sorted on the similarity score, the top 50 entries will be shown as a dataframe.
* Map View
  + This page displays a map of the U.S. showing all filtered entries, which supports zooming, as well as clicking on specific entries to display job data attributes stored in the backend.
* By State
  + This page displays a pie chart showing the state composition of filtered entries.
* By Company
  + This page displays a bar chart showing the company composition of filtered entries.
* Important Notes for Running Ngrok
  + Make sure to kill the Ngrok session then navigate to Runtime > Disconnect and Delete Runtime, this will allow another user to run it after you are done.

# Further Development

In this section, we note opportunities for growth - limitations to improve and overcome. These would involve updating both the backend and frontend.

First, for the sake of practicality, it is critical to add an automated update of the jobdata table. This would mean regularly retrieving, transforming, and appending new job postings to the stored data. There are 2 options:

1. Retrieve new API responses (tune how frequently and how many - note a request for 1000 postings gets data as far back as 3 months before requesting), compare against existing database, upload non-duplicates
2. Change API retrieval to stream new data only, upload

Next, there are some API weaknesses to address. We should analyze coordinate accuracy (some longitude/latitude values are inaccurate, or missing entirely) and skewed job listings (companies included lack in diversity (overwhelmingly Siemens from The Muse, Lyft from Adzuna)). To resolve this, we would either adjust the way we request from our current APIs, or expand to different APIs. Note that while Adzuna and The Muse have per day and per minute API request caps, they are still generally freely available, while there is currently no free access to LinkedIn, Handshake, Indeed, etc.

Importantly, the 4 description features we added still need to be improved upon to be used categorically. The OpenAI results are still messy, so we either need to thoroughly test and refine the OpenAI API requests, or find a different process. Below are some OpenAI complications:

1. 1 free account’s credits were used up after trial and error and about 4000 requests
2. per minute cap causes errors on average every 40 or so requests that need to be automatically handled
3. freely guesses, straying from original text

Finally, thinking bigger, we’d like to increase user functionality, allowing more advanced processing and matching of user input. One idea is to generate cover letters (we have shown that the following are freely scrapable: Resumegenius: 2 cover letters ([collected here](https://colab.research.google.com/drive/1SCFUkQ6R26AZWY3wt_2SEoQMCycZYMWG?usp=share_link)) and Resumecompanion, Wordhippo: adjectives commonly used in job applications and their synonyms ([collected here](https://colab.research.google.com/drive/1e4eHLKewBPx3nfAHrR1ImWtw2olXcRtR?usp=share_link), [stored here](https://drive.google.com/drive/folders/1v_o3wE-krj49hFY2v-2OWlWK-FZgFgg2?usp=share_link))). We may also consider expanding our use of Adzuna API’s top companies endpoint ([collected here](https://colab.research.google.com/drive/17s5MvE0Wu2lHC-V4o9YmwV6XUGO-OLef?usp=sharing), [stored here](https://drive.google.com/file/d/1MuobQsorjurD-Uac5L__AzbUiImLULcy/view?usp=sharing)). More generally, we’d like to identify a users’ greatest strengths and weaknesses in a particular job market segment.

# Appendix

The database where our data is stored can be accessed with the following connection:

conn\_string = 'mysql://{user}:{password}@{host}:{port}/{db}?charset={encoding}'.format(

user='team',

password='tbmFmVOE6gg=',

host = 'jsedocc7.scrc.nyu.edu',

port = 3306,

encoding = 'utf8',

db = 'Team'

)

engine = create\_engine(conn\_string)