Career Path Web Application

CSC 492: Senior Project II
Student: Annamarie Roger
Major of Study: Computer Science

Advisor: Dr. Phillip Nico
Date: June 8th, 2018

Department of Computer Science and Software Engineering
California Polytechnic State University
San Luis Obispo, CA

TABLE OF CONTENTS

BACKGROUND

The career advice that incoming college students receive leading up to their choice of major is ineffective. *One study reports that 64% of college graduates are at peace with their choice, and 32% of college graduates surveyed have never worked in a field related to their major. 36% reported wishing that they picked a different major (“*One-Third of College-Educated Workers Do Not Work in Occupations Related to Their College Major”*).* Sources of career advice range from high school counselors, family, friends, career quizzes, and personality tests. Some may be biased or outdated, lacking aggregate input from professionals established in various careers and lifestyles, or simply not tailored to those seeking career insight.

There is a need for a different approach that allows a prospective student to outline career priorities and receive suggestions drawing from a repository of reports from professionals in the field. The system will not be biased in the same way as individuals who personally know the user may be. Furthermore, the system will be periodically updated to include recent reports in order to reflect the current job market.

The system will be data-driven, and an inherent limitation is that there may be unknown biases present in the data. This limitation must be considered; the intended use of this system is not to replace existing sources of career advice. Rather, the intended use is to supplement existing sources with an accessible data-driven perspective.

REQUIREMENTS

> R1**:** The system shall provide a user interface to gauge the user's desired levels of creative freedom, social activity, income level, and independence in an occupation.

> R2: The system shall retrieve data from https://www.thecareerproject.org/ and store the data in a queryable format.

> R3: The system shall display possible occupations along with relevant occupational projections data from the Bureau of Labor Statistics' dataset.

Learning Objectives

The project requires the student to learn about:

- Data management and visualization strategies
- Building web applications
- Services and process monitoring

SYSTEM OVERVIEW

The user can use the system to gain insight into career opportunities. The following visualizations are provided by the user interface on a web application.

Visualizations:

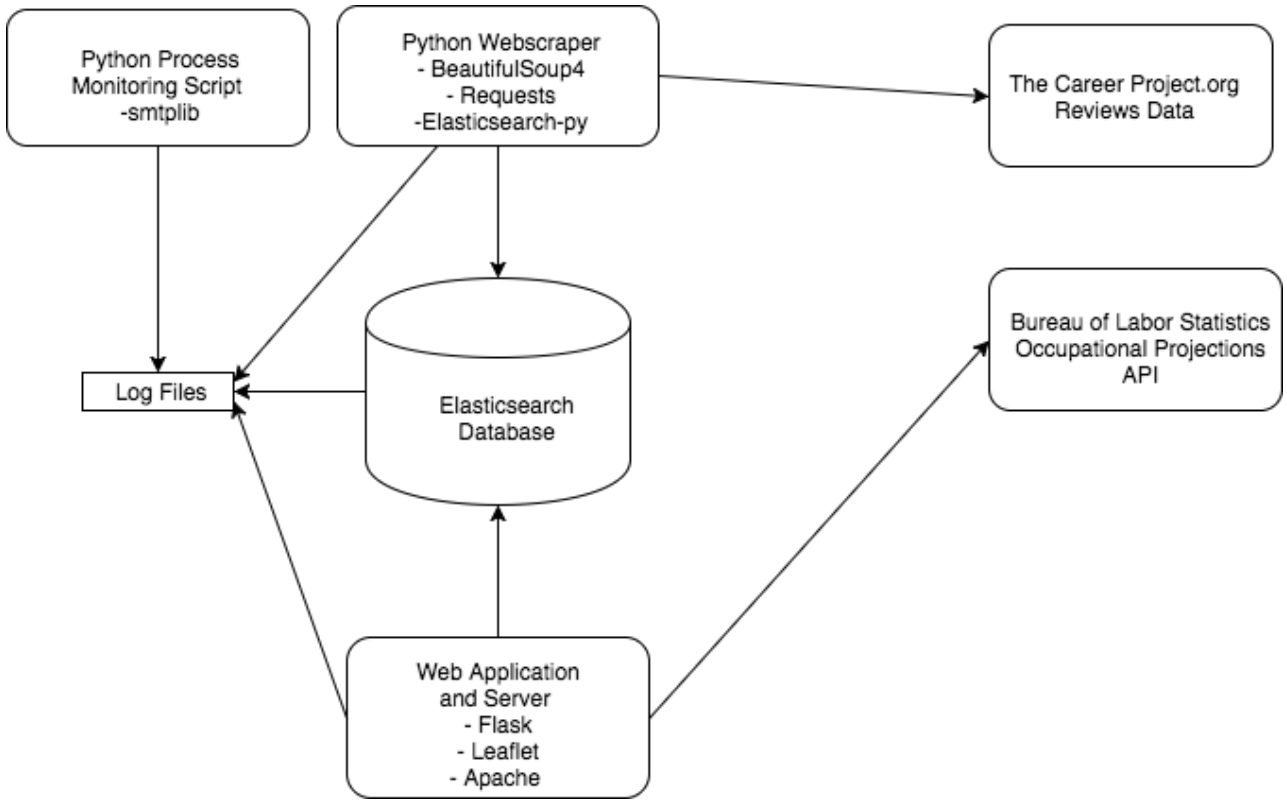
- Listed income of selected job title (e.g.: “Illustrator”) compared with other job titles pulled from stored reviews within the same field (e.g.: “Media and Arts”)
- Growth in the selected field(s)
- Map with marker of review(s)

The web application is a combination of Flask (Python code), HTML, CSS, and Javascript. Javascript code pulls data directly from a dataset and API provided by the Bureau of Labor Statistics as well as data stored in the Elasticsearch database’s REST API.

The data stored in Elasticsearch is semi-structured textual data of anecdotal reviews. The reviews are written by professionals in a variety of fields about their careers. The data is scraped using Python code and libraries from www.TheCareerProject.org.

For maintenance and debugging, unittests are created for the Python web-scraper script. Additionally, another Python script periodically reviews log files and sends email alerts with smtplib upon encountering error messages.

SYSTEM ARCHITECTURE



COMPONENT DESIGN

Python is the primary language used. The Python community is substantial and provides sufficient documentation on libraries for creating web applications performing data science.

> Web-Scraper (Requests, BeautifulSoup4, crontab)

- The web-scraper is a Python script. Requests and BeautifulSoup4 are Python libraries that are used to gather the text within HTML tags on http://thecareerproject.org.
- The web-scraper script is scheduled to run once each day using linux crontab.
- The data gathered by the web-scraper is stored in json form into the Elasticsearch database in order to fulfill Requirement 2.

> Database (REST API, elasticsearch-py)

- Elasticsearch is a non-relational database that is optimized to perform quickly on text-based queries.
- Elasticsearch has a REST API and queries can be made using GET request or from Python scripts using an available Python wrapper. This fulfills Requirement 2.
- Elasticsearch is run as a service.
- Note that this project will not involve database backups because the data can be recovered by re-running the Python web scraper script.

> Web Application and Server (Flask, Python, Javascript, Leaflet, Leaflet-geosearch, Apache httpd)

- The Flask web application has a simple web page. It prompts the user to provide a set of priorities. The priorities are used to form queries for the elasticsearch database and display a table of results. This fulfills Requirements 1 and 3.
- Map visualizations are made with leaflet.js. The leaflet-geosearch library provides the capability to turn addresses into coordinates that can be used to create markers on leaflet maps. Leaflet-geosearch will be used with Open Street Map, which is open sourced.
- Statistics on selected occupations are queried from the Bureau of Labor Statistics' API to display data from the Occupational Projections dataset. This fulfills Requirement 3.
- Apache httpd handles connection requests. Apache httpd has the capability to process dynamic content without requiring external software.

> Process Monitoring (smtplib)

- A Python script runs as a service on the server. The script reads the log files generated by the web-scraper and the elasticsearch database.
- Smtplib is used to alert on any error reports found in the log files via email.

> Testing (unittest)

- The Python unittest library provides a testsuite framework and test runners to efficiently organize unit tests.

DATA DESIGN

Following are the data sources used for this project.

I. The Career Project
Website:
https://www.thecareerproject.org

Description:
The Career Project contains approximately 1700 anecdotal reviews. Each sample of semi-structured textual data as a collection of answers to a set group of questions.
Example: https://www.thecareerproject.org/job-profile/4951/
The reviews can be distinguished by these answers, including:

Format:
Short textual answers: job title, field, age, gender, location, yearly income.

Questions are divided into sections:
- Background (job title, company description, job description, compensation, satisfaction of compensation)
- Work environment (level of collaboration and supervision, reviewer's career priorities)
- Advice to find a similar job (education/training, social skills, technical skills)
- How the reviewer landed the job (where to find the job, application process)
- Future (career goals, satisfaction, etc.)

Field	# of Reviews
Business/Management	279
Computers/IT	46
Education	107
Engineering	39
Finance	133
Gov/Law Enforcement	28
HealthCare/Allied Health	81
HealthCare/Medical	31
Nursing	45
Hospitality/Transportation	63
Human Services	54
Legal Services	51
Marketing and Sales	106
Media and Art	68
Science	36
Skilled Trades/Construction	32
Uncategorized	539
Total Reviews	1738

The reviews are organized by field as shown in the figure to the right.

II. The Bureau of Labor Statistics

Website:

https://data.bls.gov/projections/occupationProj

Description:

The Bureau of Labor Statistics provides a dataset, “Employment Projections Data” for 819 occupations with statistics from 2016 alongside those projected for 2026. The statistics are provided for both occupations and industries.

Format:

Numerical values for Typical on-the-job-training, Typical Education Needed, Occupation Openings in 2016, Occupational Openings Projected in 2026, Occupational Openings from 2016-2026 on various occupations.

EVALUATION

The requirements stated in section II are used as a basis for evaluating the results.

Evaluation of Implementation: R1

> R1**:** The system shall provide a user interface to gauge the user's desired levels of creative freedom, social activity, income level, and independence in an occupation.

The current state is a Flask webpage on localhost with a redirect option from the home page to access the search page. On the search page, a SearchForm with two StringField objects and one SubmitField object is in effect. The user may use these to submit a search query including a job and a location of interest. These two options are only a subset of the options specified in R1 and further progress on the project would strive to meet this requirement fully.

During the implementation phase, thecareerproject.org sites which are the source of the textual review data are no longer up and functioning. They display an HTTP server 500 error code. In effect, the full data of job reviews is not available to be queried.

Fortunately, several samples of the data were already queried and stored into the elasticsearch database that would permit the continuation of developing a proof of concept of this project. Still, this imposed a restriction on the project's potential to align completely with the design.

When developing the user interface, it became clear that it would be useful for the user to perform searches against specific questions from each review. This would be made possible by remapping the elasticsearch database index so that the reviews' answers are stored in distinct fields corresponding to each question rather than keeping them all together as part of an "answers" field.

Ordinarily, reorganizing the mapping of the database would be as simple as deleting the index, creating a new index with an updated mapping, and rerunning the website scraper Python script to gather review data for storage. Instead, additional code is needed to transfer the data from the old index into a new one that is customized with an updated mapping. This is a known tradeoff of elasticsearch and databases that use indices for searching.

The user interface is in very early stages as the functionality of the database and basic website features are prioritized over UI. A simple search form is presented to the user along with answers to those values. Further improvements that appeal to aesthetics and interactivity as detailed in the system overview have been delayed. These include adding toggles to allow the user to indicate additional preferences, even if the query doesn't yet support it as well as a map to visually indicate the locations of reviewers' jobs.

Evaluation of Implementation: R2

> R2: The system shall retrieve data from https://www.thecareerproject.org/ and store the data in a queryable format.

The current state fulfills this requirement. The Python web-scraper script uses the BeautifulSoup and requests python library to iterate through all job reviews posted on thecareerproject.org website when it is fully online. The script puts this data into the elasticsearch database. The current mapping sufficiently searches the data and provides relevant answers within a negligible amount of time.

Mappings are json documents where anticipated fields are labeled with the data type that best describes them. Elasticsearch has a dynamic mapping that lets system function even when no explicit mappings are provided. In our case, the review answers strings are by default already being interpreted as one large string. A match query is able to search this large string. It will not be as efficient as narrowing the search to match against only a specific subset of the reviews' answers. For example, searching for a job title name against all reviews' answer #2 data, where the job title information is most likely to be recorded will be faster than searching for the job title name against the set of all answers from every review.

It's possible to change the mapping so that in each review's answers are interpreted as discrete strings, (i.e.: Answer 1, Answer 2, …, Answer N). In this case, Elasticsearch's multi_match query would allow this nuanced querying. The process for this is detailed above and will be done in future work.

Evaluation of Implementation: R3

> R3: The system shall display possible occupations along with relevant occupational projections data from the Bureau of Labor Statistics' (BLS) dataset.

The current state fulfills this requirement. Upon a user's request to search, the review data in the elasticsearch database and data from the BLS' Occupations Projections dataset that are relevant to the query are displayed on the Flask webpage below the search feature.

The Bureau of Labor Statistics' API uses a system of codes to represent the parameters needed to form a query to their data. For example, the user's specified location must be translated to a predefined area_code. Likewise, the job to search for that the user specifies must be matched to an occupation_code and so forth. Further specifications into the BLS' code system are cited.

With six different codes to look up, at least six files must be consulted to form a query. Once the query is formed, the script may send it within an HTTP GET request to the Bureau of Labor Statistics' server.

There are 1000+ records in the BLS' Occupations Projections dataset. This number of records is manageable and can be stored on disk without complication. To avoid the complicated process of forming and sending queries with the BLS' API to the BLS' server, it was decided that the full dataset may be downloaded and accessed directly.

It may seem that there is a tradeoff: the data is not updated in real-time. This issue would be mitigated by downloading the dataset each day to account for updates, but is not necessary because content of the data is related to the time of creation. For example, the number of jobs in specific field reported in 2015 is a fixed value that should not need to be updated.

<u>Evaluation of Engineering Practices:</u>

> Tests
The current state of this project has few tests. Unit tests, system tests, and integration tests are needed to develop any proof of concept into a reliable system. Development was done solely on localhost with the intention of winnowing the number of possible bugs. The flipside of this choice is that tests will prove to be especially necessary and useful when migrating a more mature version of the system to an external server.

> Security
User input must be escaped and validated to mitigate cross-site scripting attacks.

The configuration file holds a secret key and needs to be secured so that the system is not vulnerable to attacks.

> Logs

Another strategy that supports ease of system maintenance is to keep a logging system. The current state records logs on disk which is localhost. To avoid abusing the memory limits of the system, a script to periodically purge these logs will be needed.

> User Engagement

The data visualization needs work as a systems' effectiveness could reasonably be evaluated based on the users' engagement levels.

> Timeline

A successful project meets all requirements. Detail-oriented and realistic planning can be advantageous techniques for achieving this goal. While not everything can always be foreseen, this project would have benefitted from having a more detailed timeline.

CONCLUSIONS

This chapter should summarize the overall experience of the project and potentially suggest possible directions for future work. I tell my students, "The final report should be the document you wish you had had in your hands when you began this project."

> Future work on this project will involve:

- Improved user interface as detailed in the system overview
- Improved Elasticsearch mapping to allow for more efficient queries
- Schedule the python web scraper into a recurring task with Linux crontab
- Periodically purge logs to avoid consuming too much memory
- Launch the system on a server separate from localhost
- Securing the configuration file

> Lessons Learned:

- Prioritize creating the "right" database mapping as early as possible because it may otherwise become the bottleneck of the project.

- Create a copy of anything that the project largely depends on as soon as possible, (ie: external data sources). This will allow the outcome of the project to be independent of the external source's reliability.

- Plan and describe achievable milestones, down to the small details. This may help to:
    - Maintain consistent progress
    - Devise a project of realistic size
    - Keep the target goals in sight

- Work with a team. This might help to maintain consistent progress and generate inspiration when tracking down particularly persistent bugs.

I would like to thank my advisor, Dr. Nico, for supporting the development of this project and my education here at Cal Poly. It was intrinsically rewarding to explore and experience web development, an area in which I have little prior experience. The process of preparing a project and taking steps to implement the plan was a learning experience in and of itself. I am thrilled to have further solidified a perspective that nothing is out of reach without careful strategy, keen ears, and patient persistence.

WORKS CITED

"EP News Releases." *U.S. Bureau of Labor Statistics*, U.S. Bureau of Labor Statistics, www.bls.gov/emp/ep_table_107.htm#top.

Gormley, Clinton. "Changing Mapping with Zero Downtime." *Open Source Search & Analytics · Elasticsearch*, 16 Apr. 2018, www.elastic.co/blog/changing-mapping-with-zero-downtime.

"One-Third of College-Educated Workers Do Not Work in Occupations Related to Their College Major." *One-Third of College-Educated Workers Do Not Work in Occupations Related to Their College Major - CareerBuilder*, www.careerbuilder.com/share/aboutus/pressreleasesdetail.aspx?sd=11%2F14%2F2013&siteid=cbpr&sc_cmp1=cb_pr790_&id=pr790&ed=12%2F31%2F2013.

Point, Tutorials. "Python Sending Email using SMTP." *Www.tutorialspoint.com*, Tutorials Point, 5 Oct. 2017, www.tutorialspoint.com/python/python_sending_email.htm.

"Python Elasticsearch Client." *Python Elasticsearch Client — Elasticsearch 6.1.0.Dev documentation*, elasticsearch-py.readthedocs.io/en/master/.

"Linux crontab command help and examples." *Computer Hope*, 19 Oct. 2017, www.computerhope.com/unix/ucrontab.htm.

"Employment Data Definitions." *U.S. Bureau of Labor Statistics*, U.S. Bureau of Labor Statistics, www.bls.gov/emp/ep_nem_definitions.htm.

DigitalOcean. "Contents." *Apache vs Nginx: Practical Considerations | DigitalOcean*, DigitalOcean, 18 July 2017, www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations.

"Refine Your Results." *U.S. Bureau of Labor Statistics*, U.S. Bureau of Labor Statistics, data.bls.gov/projections/occupationProj.

Smeijer. "Smeijer/Leaflet-Geosearch." *GitHub*, 2 Oct. 2017, github.com/smeijer/leaflet-geosearch.

"Documentation - Leaflet - a JavaScript library for interactive maps." *Leaflet*, leafletjs.com/reference-1.2.0.html.