Alexander Brown, Ruijie Rao, Ahmed Alsalim
DSCI551 Spring 2022
Project Report

Labor Statistics Dashboard for Data-Informed Career Path Decisions

Motivation

Right now, a record number of people are leaving their jobs and changing careers due to disruptions caused by the COVID-19 pandemic. According to Congressional Research Service, the unemployment rate in the US reached 14.8%—the highest rate observed since data collection began in 1948—in April 2020. Moreover, according to the latest CNBC jobs report, the number of people employed part-time for economic reasons, but would rather be working full-time, increased to 6.7 million in October after declining for five months. Meanwhile, choosing a new career is one of the most impactful decisions a person can make. It often takes years of careful planning and enormous costs. Unfortunately, many people often make very important decisions about which career they want to pursue or how to prepare for such a career, without a thorough knowledge of the current or future labor market. Researchers found some factors of Career Decision Difficulty (CDD), such as employment status, lack of accurate information about careers, gender, and more. (Abdulfattah and Nizar). One additional complication to this issue is that much of the information that could be helpful in making career decisions, such as average salary, unemployment rate, worker hazards, turnover rate, and educational costs, is often hidden behind convoluted government or corporate datahubs and APIs.

Topic

For our project, we worked on a way to ease this problem by creating a user-friendly dashboard to explore important labor statistics based on career interests, educational level, and geographic regions. We feel that providing this information on a large scale could help ease some of the disruption caused by the pandemic and our changing labor markets. Our primary data source for this project is the U.S. Government Bureau of Labor Statistics. We synthesized and joined information from various datasets such as the Occupational Employment and Wage Statistics, Current Population Survey, Modeled Wage Estimates, Occupational Requirements Survey, and Employment Projections. We also feel that on a personal level, having access to important data about the labor market, could be hugely beneficial and stress relieving when trying to navigate a new career or make a career change. We focus on monthly data at the state level for 2019-2020 to also reflect the potential disruptions caused by the COVID-19 pandemic. However, much of the data comes before the pandemic so the disruptions may not be as pronounced. We initially thought the 2021 data would be released prior to the end of the semester but unfortunately, it is running behind schedule and has yet to be released.

Dashboard URL

Career Dashboard: http://13.57.225.54:5050/

Data Sources

- US Bureau of Labor Statistics: https://www.bls.gov/oes/tables.htm
    - This government website includes a variety of labor market data including salary and employment rate for most positions, entry-level education requirements, retention rates, and other attributes. Moreover, the data is recorded annually from 1988 to 2021, which allows us to do some pattern analysis with huge time bounds. Since it is a government primary dataset, we believe that the data will be cleaner and more trustworthy than other public resources. As stated previously, the data is broken down into separate datasets based on the topic. Examples include Occupational Employment and Wage Statistics, Current Population Survey, Modeled Wage Estimates, Occupational Requirements Survey, and Employment Projection.

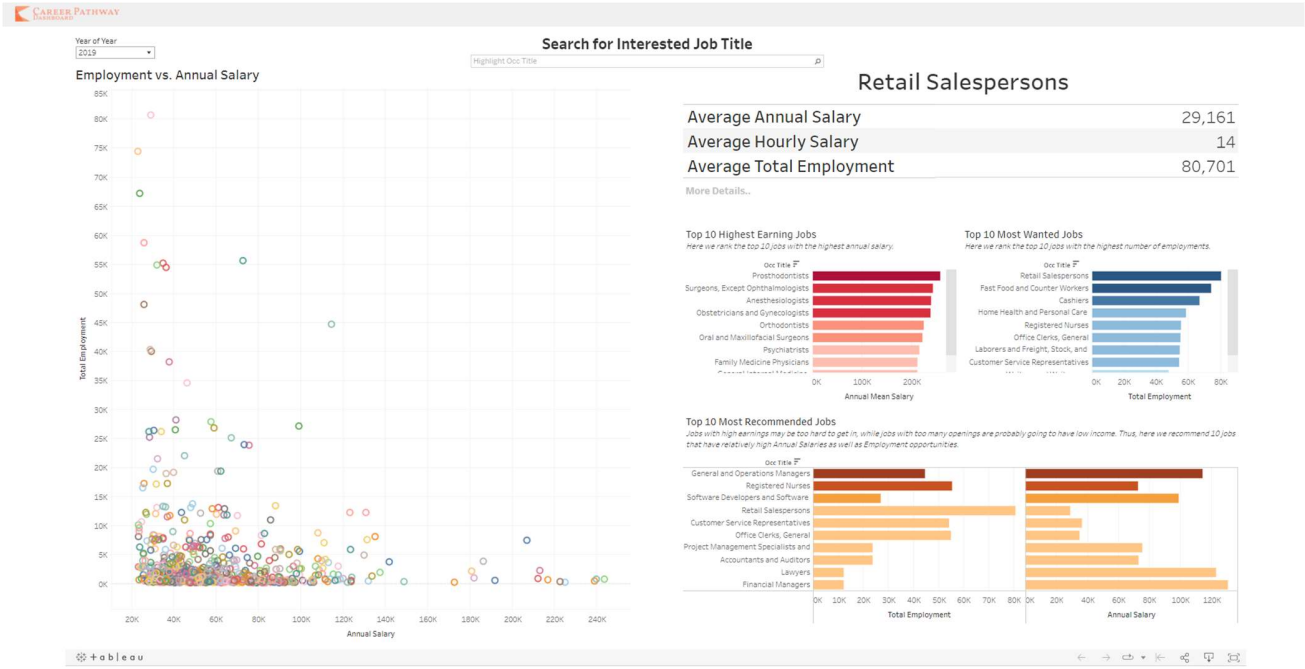
App Architecture

For this project, we have chosen Python Flask as our backend framework due to it being lightweight, simple to learn, and has a broad community of users. The app is composed of 2 routing pages: **Career Overview** and **Career Details**.

- **Career Overview** is an embedded tableau page that visualizes and ranks every career we have in our database according to their employment and income data. The aim of this page is to show the current labor market in an interesting and colorful way, while also enabling users to efficiently compare a large number of jobs bi-dimensionally. After visiting this page, we would like the users to have a better idea of what jobs are popular, what jobs have the best earnings, and what jobs should they investigate more on. By highlighting a specific job, the user can then click on "More Details" button to explore more data about it on **Career Details** page.
- **Career Details** displays a brief description, detailed statistics, and a variety of charts of the chosen career. Charts include choropleth maps for employment numbers and salary, Line plots for employment/salary trends, and histograms for wage distributions across a profession. The plots were all developed using Python Plotly. The aim of this page is to allow users to dig deeper into any career by delivering to them valuable information like employment ratio by state and change in income through years. We hope our users could have a much better understanding on specific careers that they are interested in and be able to make better decisions after visiting this page of our website.

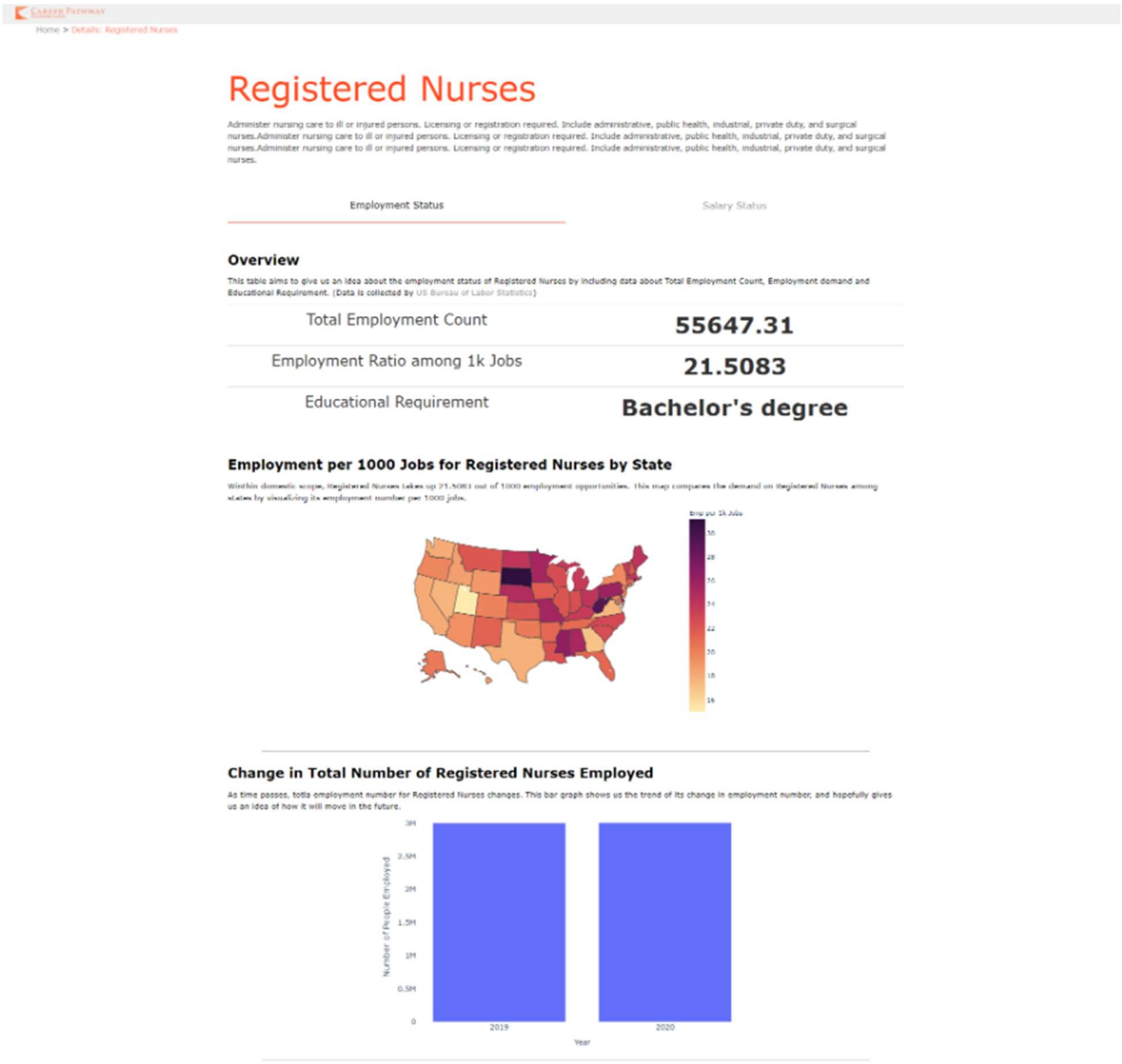
The very general structure of the app consists of a primary routing python file that handles the routing between the two pages. Depending on the URL, different functions have triggered that call up different HTML/CSS functions. Each page has a static outline and that is completed dynamically with either a Tableau or Plotly plot using JavaScript calls.

Screenshots



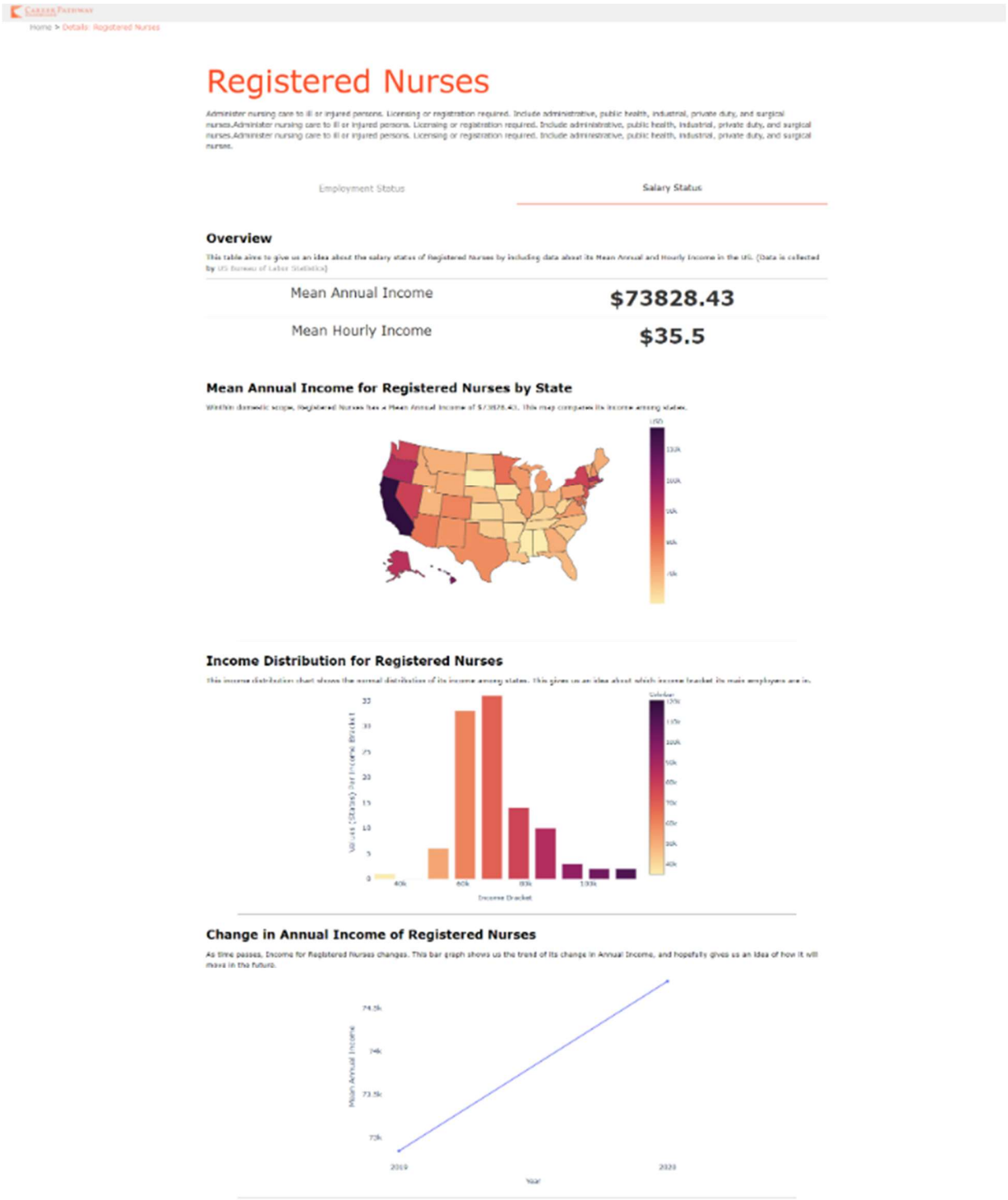
▲*Page:* ***Career Overview***

- *On the left hand side of the page, there is a large scatter graph that plots all the careers on their employment rate and annual salary.*
- *On the right hand side of the page:*
    - *A table showing brief information on salary and employment of the highlighted career.*
    - *A rank on salary in red and a rank on employment rate in blue.*
    - *A rank on salary×employment in gold*



▲*Page:* ***Career Details-Employment***

- *Title shows the name of career title.*
- *Career Description.*
- *Overview Table of National Data: Employment Count, Employment Ratio and Education Required.*
- *Map plot showing employment ratio in every states.*
- *Bar plot showing change in national employment of this career across years.*



▲*Page:* ***Career Details-Salary***

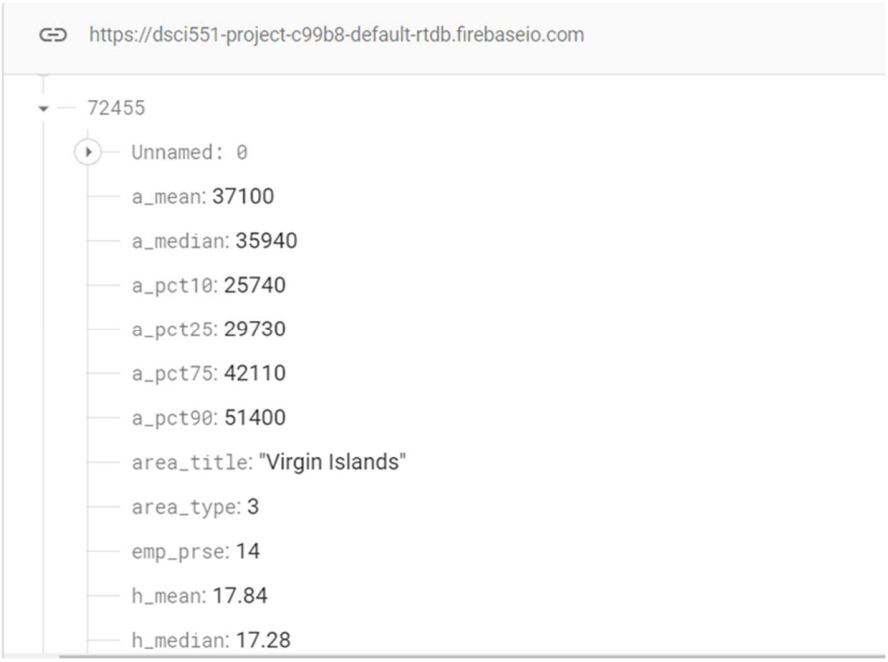
- *Title shows the name of career title.*
- *Career Description.*

- *Overview Table of National Data: Mean Annual Income and Mean Hourly Income*
- *Map plot showing mean annual income in every states.*
- *Distribution plot showing income distribution.*
- *Line plot showing change in annual income of this career across years.*

Data Flow

The data was downloaded from the BLS website data portal as raw CSVs. We downloaded three primary datasets Occupational Employment and Wage Statistics (OEWS), Modeled Wage Estimates, and Occupational Requirement Surveys. The data was then cleaned and prepared in Python/Pandas using Google Colab as a platform. The preparation involved manually dropping unnecessary columns, joining the different datasets on a primary key (OCC code), removing missing values, imputing other partially missing values, and formatting data strings. Once prepared the data was saved back to one master CSV and uploaded to Google Drive and Firebase Realtime Database (via JSON). Tableau imports real time data from the CSV in our Google Drive and then makes the dashboard which is displayed on **Career Overview** page.

Firebase is a common NoSQL database that is used to provide backend support to applications and websites. We used a Firebase Realtime Database to store and sync our data from a cleaned dataset. The database supports RESTFUL API calls so that we can access the data quickly and dynamically from within the app rather than storing our data statically on an EC2. It also enables data to be quickly added without changes to the schema, which we needed just in case the 2021 data had a slightly different structure. The data in the database is stored in the JSON (ish) format which has a nested structure and can be represented using a tree. The dashboard in the Firebase dashboard is simply a visual representation of a JSON database tree. Our Firebase database will contain key-value pairs for each record. The keys are the count numbers starting from 0 to 72,466 and the values are a nested object containing complete information about the Career Details such as a_mean, a_median, area_title, area_type, occ_code, occ_title, tot_employment, year, etc.



Once the data was organized within a Firebase, we wrote RESTFUL requests using Python to query the Firebase data by specific occupation titles according to user input and read them into a Dataframe on the backend for visualization. The data from the Firebase requests was structured with the different occupation attributes (employment number, requirements, salary, year, etc) as columns and the state-level information as rows. We wrote a number of separate visualization functions using Plotly, that take the occupational Dataframe as an input and return a JSON encoding of the visualization that can be sent to the **Career Details** page and displayed using Plotly Javascript. Some of the visualizations required aggregation across states which was again, completed on the backend using Pandas.

Challenges Faced

Cleaning data

- There were quite a few missing values to be dealt with, one example was the dataset contained '*', '**'for missing values and '#' for values that were above the set range. To deal with the '*' missing values we set them to NaNs and ignored them in aggregation functions. Large gaps between salaries across states (aggregation errors)
- The upper limit on salaries (caps 200K)
- Values marked with '#' indicated that the value was outside the range of the dataset, and therefore should be treated as outliers. One example of this would be the occupation (CEO), in which the values vary wildly. Instead of dripping the rows, the #s were imputed to the max/and min values for a given feature. This was done so that aggregations wouldn't be massively skewed by outliers and we wouldn't lose information.

Limited time/space in Firebase

- Adding additional data to firebase was one of the major challenges as it has limited free space. The same issue will be encountered when the data is expanded to include more years than just 2019-2020.
- The time it takes to add data to Firebase was significant. Loading more than 72,000 for a single year of entries to the firebase takes a long time. Likewise, updating and adding entries took time.

Plotting (Tableu/Plotly)

- Tableau Public is the tool we found that integrates almost perfectly with web apps structures. As a result of its huge potential, learning to use it has been a great challenge. This includes designing the dashboard, considering the best visualizations to use for different purposes, and inserting user interactions. The most challenging aspect was figuring out how to link the user interaction on the tableau interface with the web app backend. The solution was to add a hyperlink button on the dashboard which directs to the routing url, which includes the string used for querying the Firebase. The backend receives the parameter from the URL and then filters the data accordingly.
- Another difficulty was on linking the second page to the first page. As you can see on the screenshots, the first page is a dashboard for all the careers, while the second page is about details of a certain career that the user chose. Tableau is not convenient on doing filtering and computing

data, so we need another plotting tool. Thus, we chose Plotly which integrates well with Python Flask and also provides a range of plots as well as interactivities.

- We also ran into a few issues with plotting using Plotly. The main issues plotting the changes in values (income, employment) between 2019 and 2020 for certain states due to missing values in one or the other year. To correct for this we had to filter the plots based on the states/years that don't have missing values.

Deploying Webapp

- Backend routing using flask has been simple, but frontend development is a new challenge. I have never fully designed AND developed a website frontend before, so I have been learning design thinking and CSS/HTML knowledge during the first phase of our project. I used Figma with my mock/flow designs as well as static image design including website icon and navigation bar.
- Although I have experience with webapp deployment on Azure, Amazon AWS is another thing. I have spent some time on researching guides and tutorials, and also with the help from lecture notes, I was able to upload my scripts on to my EC2 instance and build up the server really fast.

Reflection on Learning Experience

Alexander Brown

My portion of the tasks included data acquisition, cleaning, and backend visualization. The data cleaning portion of the project was fairly straightforward and I ran into many of the same issues I have encountered in prior projects. However, one unique aspect of the cleaning process was in dealing with data that was outside the numerical range of the features (incomes above 250K, etc). In this case, simple deletion or imputation to the mean would likely remove important data. We elected to impute to the max, or min, depending on which end of the range. This reminded me how there really isn't one way to deal with messy data, and often times decisions can come down to simple value judgments around what information needs to be preserved and what does not.

In the visualization portion of the project, the one learning experience was in building function scripts that were built to a very specific specification (e.g. being able to be converted to JSON format, specific return types, and function parameters), any deviation from specifications could potentially break the front end portion. For this, I had to use lower-level Ploty graph objects rather than the usually Plotly express objects. I also had to make sure I structured the parameters and returned objects from the functions so that they were compatible with the app and ran quickly.

Ruijie Rao

Data Visualization has always been my interest but I have never digged deep into it. This project gives me a motivation to learn about data viz tools like tableau and plotly, as well as how to embed them into web applications so that it can be seen by not only me but also everyone. During the process, I have learned about balancing effectiveness and beauty of visualizations by choosing the best type of plots with suitable color schemes. I have also learned about building a dashboard that interacts with the users through filters, searches, highlighting and buttons.

On the other hand, deploying our Flask app on EC2 instance has also been a new experience, which took me lots of time and effort. Learning about using gunicorn3 on the instance has been confusing and sometimes frustrating. However, being able to deploy our app onto the internet is really a rewarding experience and I believe this learning process has broaden my technical capabilities on another dimension.

Ahmed Alsalim

The first part of the project is completing data preprocessing and data exploration which include data cleaning, outlier treatment, data categorization, and data creation. I used to perform this task manually on a small dataset. However, in this project, we are using a big data set consisting of more than 72,000 records. So, while working on this project I learned how to use python/panda to prepare and clean this amount of data quickly. Also, using the tool to identify and treat missing values and outliers, convert values to the correct format, and select the needed features.

In the Database portion, we used the NoSQL database firebase to upload the cleaned dataset. First, we have converted the file to JSON format using the python/panda module as firebase cannot read csv files. Then, upload the data to firebase using CURL commands. Also, sat the rules and granted access to my team members to manipulate the data. It wasn't easy to upload a large amount of data which requires a longer time than usual.

Team Members and Responsibilities

See above
Ruijie Rao (front end, backend routing, deployment)
Alexander Brown (data acquisition, backend visualization scripts/visualization)
Ahmed Alsalim (data cleaning/merging/database setup/RESTFUL requests)

References

Jones, Alan. (2021). “Web Visualization with Plotly and Flask.” *Medium (Towards Data Science).* URL: https://towardsdatascience.com/web-visualization-with-plotly-and-flask-3660abf9c946

Falk, Gene., Romero, Paul D., Nicchitta, Isaac A., Nyhof, Emma C. (2021). “Unemployment Rates During the COVID-19 Pandemic.” *Congressional Research Service.* R46554. https://sgp.fas.org/crs/misc/R46554.pdf

Lui, Jennefer. (2020). “Millions of Workers Remain Underemployed as Visur Surges Prompt Closures.” *CNBC.* https://www.cnbc.com/2020/11/19/millions-of-workers-underemployed-as-virus-surges-prompt-closures.html

Yaghi, Abdulfattah., Alabed, Nizar. (2021).“Career decision-making difficulties among university students: does employment status matter?” *Higher Education, Skills and Work-Based Learning* https://www.emerald.com/insight/content/doi/10.1108/HESWBL-07-2020-0149/full/html

Dashboard URL

Career Dashboard: http://13.57.225.54:5050/

Code Folder

https://drive.google.com/drive/folders/1GIUjvz66coDML1Q9gHkQ8UeEEiqBirKi?usp=sharing