Phase 2: Infrastructure and Technology Stack

In order to construct my data lake, microservices, and web app api a variety of core technologies were required. To start, my data lake was powered by a PostgreSQL data engine. This relational database engine enabled the ingestion of data from a variety of data sets from the City of Chicago Data Portal. Additionally the PgAdmin4 user interface which supports PostgreSQL was convenient for visualizing data ingestion real time. The fundamental technology platform used to develop the microservices for this application was the Python Programming Language. While many other libraries were required for successful execution of the microservice tasks, Python was the programming language that propped everything up. The third core technology used in this project was Flask web framework to create the http api for allowing users to access the microservice functionality. Lastly, while not required for the execution of the application, Visual Studio Code was the IDE used for all development which was helpful in the construction, debugging, and testing phases of the project.

In addition to the core technologies used in the design and execution of this project, there were other considerably important technology packages that were enabled to give this app and microservices the functionality required:

Pandas v 1.1.3: Pandas was the core package used for handling data within python. This dataframe library is a staple in the python arsenal and provided the ability to store, manipulate, and export the data utilized within this project.

Pyscopg2 v 2.8.6: Pyscopg2 was the package leveraged in order to connect and interact with PostgreSQL via Python code. This package facilitated creating databases and data tables in PostgreSQL as well as reading and writing the data within those tables.

Folium v 0.12.1: Folium was a package used to visualize Covid-19 and taxi data. Folium is a library that allows for users to create maps paired with data for interactive visualizations.

Geopandas v 0.8.1: Geopandas is a package very similar to Pandas. The only difference is that geopandas enables users to interact with geojson type data within familiar dataframe objects.

Sodapy v 2.1.0: Sodapy is the api package that allowed for the connection and extraction of data from the City of Chicago Data Portal.

Shapely v 1.7.1: This package was vital in connecting community areas, community names, and zip codes. This package enabled the ability to detect if certain latitudinal and longitudinal points were contained within geojson shapes (Zip Codes).

Numpy v 1.19.2: Numpy allows for mathematical manipulation of data objects within Python. This package was used to execute calculations and aggregations of the data within this application.

Sklearn v 0.23.2: Sci-kit learn is a package that gives the user access to many useful machine learning utilities. This package was used to perform linear regression of taxi trips and percent positive covid-19 tests rates for various zip codes within a given week.

Plotly v 4.14.1: Plotly is a data visualization package that outputs interactive charts and graphs. This package was used for a variety of reasons to make the data output dynamic for users of the application.

Prophet v 1.0.1: The Meta Prophet package enables users to make time-series forecasts. This package enabled the ability to monitor historical records of taxi data and forecast taxi pickups and dropoffs within various area codes in the City of Chicago.

In addition to these technologies utilized within the construction of my data lake and microservices, a number of other technologies were considered for their potential utility within this application.

A Redis data engine was considered as the data engine of choice to create the data lake required for this application. However, Redis was ultimately not chosen as I felt a relational data model better fit the inherent form the data was already in when pulled from the SODA API. This data was better suited for relational databases and tables as opposed to a hash structure of storage that Redis provides.

Golang was another programming language considered for use when building the microservices for this application. While Golang is very popular for building microservices, particularly in the cloud, this programming language was omitted from this application due to my stronger familiarity with Python.

Docker and Kubernetes clusters are another technology omitted from this application. This technology was heavily researched, similar to Golang, for use in developing and deploying this application with the various microservices. However, I ultimately decided not to containerize my application with Docker as the probability of deploying this application is low. Therefore, since this application is primarily for personal use, I chose to avoid containerization and networking which Docker and Kubernetes help facilitate.

Geopy is a Python package that helps make working with geospatial data easier. For instance, latitudinal and longitudinal data can be pulled for popular sites and landmarks, distances can be calculated between two points, and these distances even could be displayed on a map. While my application did utilize geospatial data, Geopy was omitted from development as the geospatial data needed for my microservices were provided from the City of Chicago Data Portal. Also, I felt some of the Geopy features could have misled the users of this application. For instance, when tracking individual taxi and transportation network provider trips, calculating and displaying the distance between pickup and dropoff as a straight line would be misleading. In short, the data pulled from the City of Chicago Data Portal was sufficient.

Arcgis is a package similar to Folium. Arcgis is a cloud service intended for creating and sharing interactive maps. Ultimately, I felt there was considerable overlap between Arcgis and Folium for the desired utilization and deferred to using Folium as I am more familiar with that library.

Google Maps Services for Python was also deeply researched but ultimately not utilized within my application. This library could have been used to estimate the routes taken by taxi drivers or transportation network providers. Google Maps could also be used to estimate road condition and axle traffic throughput for planning streetscaping investments for the City of Chicago. However, this library was not utilized as I felt the estimates from Google Maps would likely not reflect actual traffic data and lead to misinformed decisions from stakeholders. However, if given more time to develop this service, including and verifying the accuracy of the data provided by Google Maps is something worth considering for potential improvements.

Lastly, Tensorflow/Keras LSTM was another technology researched for this project but ultimately not used. These packages could be used to implement AI/ML models within the project. However, I preferred to work with Sci-kit learn and Prophet for performing regression and time series forecasting.