Applied Research Project

Zyp Art Gallery Social Media Dashboard Application

Final Report

Student Identification Number	300328339
Student Name	Kunal Ajaykumar Jeshang
Course	CSIS 4495-001
Semester	Summer 2022
Instructor	Prof. Stephen Chiong

Table of Contents

Abstract/Executive Summary

The goal of this project is to create a custom dashboard application to help the Zyp Art Gallery non-profit organization visualize extracted Facebook & Instagram data, and derive insights through. This project would assist Zyp Art Gallery make strategic decisions to manage its social media presence that can contribute to its overall growth. Greater growth of the organization would lead to more involvement of volunteers, charitable endeavors, general community interaction, and donations. This project would apply a combination both software development and data analytics knowledge & technical skills.

Context

Introduction

As of June 2021, I have been remotely volunteering as a part-time Data Analyst for Zyp Art Gallery. It is an art gallery based in Calmar, Alberta. As a current Post Bacclaureatte Diploma (PBD) student in Computing & Information Systems (CSIS), I became interested in becoming a volunteer to apply the skills I learnt in the classroom to solve problems and make a positive impact. I found a volunteer posting on the Volunteer Connector website for Software Developer role at Zyp and applied by sending an application form via email. After my interview was completed, the Human Resources specialist and the Analytics team leader believed I would be a better fit for the Analytics team based on my academic & professional background. Most of the work I have done so far at Zyp involved Social Media Analytics.

History

The Zyp Art Gallery is an art space created & governed by the Calmar Art Society. The president of the Calmar Art Society, Dr. Nhung Tran-Davies, both founded and named the art gallery in honour of Hank and Tillie Zyp. The Zyps played a large role in Nhung's life as they sponsored her and her family from a refugee camp to migrate Canada after the Vietnam war in 1979. The Zyps welcomed her and treated her like a daughter. Nhung was always inspired by Hank and Tillie's social activism, generosity of spirit, and dedication to the arts.

Hank and Tillie Zyp were incredible human beings during their lifetime. During the 1960s, the Zyps were one of the only families in Edmonton to open their home to indegenous children that were based in reserves so that they could attend high school. This was an amazing act of kindness and hospitality as at that time, the reserves did not have any schools and no school

buses to transport children. During the 1970s, Hank and Tillie founded two non-profit organizations (i.e., ‘Change for Children’ and ‘Rainbow of Hope for Children’) that supported communities in developing nations. From the 1980s until their passing, the Zyps continued their work in giving a voice to the voiceless by supporting local community projects that involved the Indigenous community, the poor of Edmonton's inner city, and city’s youth. Hank Zyp was considered the ‘face’ of the couple who had the ability to educate and illuminate the struggles of others with his words. Tillie Zyp was considered the ‘passion’ of the couple who focused fiercely on engaging individually with people and tirelessly managing the non-profit organizations.

Furthermore, Hank Zyp was an art teacher by profession and created many beautiful pieces of art. Nhung was always amazed by Hank’s work and was saddened that those in the surrounding community would not be able to see them. That being said, building the Zyp Art Gallery was always a passion for Dr. Nhung and it took many years until that passion became a reality. The art gallery officially opened on May 15th 2018.

Background

Based on what was explained in the previous section, I was inspired to work for the Zyp Art Gallery. Furthermore, what drew me to this organization was that it aims to remove barriers “for immigrants, new Canadians, and all people of colour who look for opportunities within the arts and other professional fields by providing real-world training and opportunities through [its] volunteer and artist programs.”

As a volunteer Data Analyst, I have been working remotely. I attend weekly Analytics team meetings via Discord with the Executive Director of the art gallery along with other Analytics team members. As aforementioned, most of the work I have been focusing on involved Social Media Analytics; in particular Facebook and Instagram. The organization is reliant on social media data to help define trends that could help grow its online presence. As Facebook stopped providing un-aggregated data due to the shift to using Facebook business, which only shows aggregated data with limited metrics, Zyp Art Gallery did not have access to reliable data. That being said, I decided to use Python to extract, transform, and load Facebook & Instagram posts, page insights, and demographics data with the help of Facebook Graph API as well as Google Drive & Google Sheets API into both CSV & Google Sheets files for the purpose of deriving meaningful insights in the form of PowerBI & Google Data Studio reports. My contribution has been very valuable to Zyp Art Gallery as the data I provided has helped them to better understand its online audience, in turn, help it make sound decisions to grow as an organization. If the exposure of the organization grows, so too will its donations and funding that can be further used for endeavors that can help provide a voice to the voiceless.

Statement of the Problem

As a Data Analyst for the Alberta-based organization, Zyp Art Gallery, I have been working on extracting social media data to help derive insights. I focused solely on the extraction of the

social media data but not visualization. Thus, I now want to help the organization to visualize the extracted data. Deriving insights from social media data will allow the organization to make more strategic decisions to acquire more donations and make a greater impact on its surrounding community.

It is common knowledge that there are many specialized softwares used to help visualize data such as Microsoft Power BI and Tableau. As it is a non-profit organization, Zyp Art Gallery does not have the funds to acquire a Microsoft Power Suite or Tableau license. Power BI is free but without the license, we do not have the ability to use Power BI service to share dashboards to other members/volunteers of the organization. Tableau may have a free public version but the dashboards are saved online (NOT locally), and the published dashboard's community portal is not known to keep sensitive data safe. It is challenging to teach non-tech savvy people how to install and use these softwares. Also, not every volunteer has a powerful computer running the appropriate operating system to run these softwares. Google Data Studio was considered as a possible tool by the organization due to its use of the GSuite for email correspondance, document storage & editing, and virtual meetings. Despite being free and intertwined with GSuite, the platform was not powerful enough to process, aggregate, and analyze our social media datasets which lead to errors beings thrown.

Even in the perspective of a tech savvy individual that has the appropriate technology to run specialized softwares, pre-exisiting dashboard softwares tend to have some operational drawbacks. Below is a list of some of them.

- Limited interactivity of components (to a certain degree)
- Fixed dimensions which does not allow the dashboard to have a more organic& free-flowing design
- Runs slow or is incompatible with older devices
- Free versions have less features to best-assist the needs of the organization (including a non-profit)

Suggested Solution

Thus, to best address the problem, the ideal way to assist Zyp Gallery to be able to visualize and derive social media insights is to construct a custom dashboard application that is free, accessible, and easy-to-use.

The technologies stated below would be used to create a custom multi-page dashboard web application:

- Programming language = Python 3.8.9+
- IDE = Microsoft VS Code
    - Development = Python scripts in a multi-layer file structure
    - Prototyping of both analysis and visualization = Jupyter Notebook
- Important packages = gspread, pandas, datetime, numpy, plotly, dash, gunicorn, flask

- Data source = Social Media data files in CSV file format in Google Sheets ~~file format from organization's Google Drive (accessible via Google Drive & Google Sheets API using a Google Service Account private key)~~
    - Social media extraction code is separate from project but necessary to get up-to-date social media data
- Version control = Git
- Hosting & Deployment = GitHub & Heroku
    - Performing first time deployment using PyCharm Community Edition

The implementing organisation

The Zyp Art Gallery organization is an open minded organization willing to support its volunteers for the purpose of progressing its endeavors. Dynamic and creative problem solving is encouraged provided that there is little to no financial expenditure. This may not be ideal but is a reality that is faced by all non-profit organizations; including Zyp Art Gallery.

As touched upon in earlier sections, the Analytics team and the executive director were open to the prospect of extracting Facebook & Instagram data with the help of Facebook Graph API and Python programming. This was both a necessary and free solution due to Facebook not allowing instant download of un-aggregated data because of its intent to transition to Facebook Business which limits data transparency by only showing aggregated data with limited metrics. The executive director was open to me setting up a Facebook for Developers account and creating an app dedicated to Zyp Art Gallery within Facebook for Developers. The organization's privacy policy was required for final approval of use to gain full access to all metrics. After that, it was a matter of generating an extended access token to then be used with a Python script to make a request to Facebook Graph API, extract the data which is in an unstructured JSON format, loop through & clean it so that it is in a pandas dataframe, and export it into a CSV file. The executive director was also open to me setting up Google Drive and Google Sheets API via the Google Cloud Console so that I can apply the GSpread package to instantaneously take the contents of social media data files in the CSV format and append to a corresponding-existing Google Sheets file based on spreadsheet IDs. All of the aforementioned did not involve any financial expenditure and was welcomed by the Analytics team's workflow. The Python script is run on a weekly basis, and extracts as of the last most recent date on the CSV files.

In regards to the custom dashboard application, the extracted Facebook & Instagram data would be imperative to it. Zyp Art Gallery's executive director and Analytics teams, as well as Marketing teams would be helpful to further define what would be the best way to visualize posts, page insights, and demographics data for Facebook & Instagram. As the aforementioned in the past requested the extraction of specific types of insights data, they would ideally assist in the visualization direction as well. Their involvement is clinical because the custom dashboard application would be utilized by them. I have weekly Analytics team meetings with them on Thursdays at 5:00PM. I would ask for constructive feedback during the meetings. As I am the only individual on the team with a programming background, I would not be assisted in terms of

technical execution of the project. My interaction with the Analytics team in the context of this project would as if I am attempting to create & deliver a product to them as my client.

Project aims

Logforme Matrix:

Component	Description
Project Goal a general aim that should explain what the core problem is and why the project is important, i.e. what the long-term benefits to the target group are	At the moment, Zyp Art Gallery does not have a long term & sustainable solution to derive social media (i.e., Facebook & Instagram) insights in the form of easy-to-understand visualizations. Thus, the aim of this project is to construct a custom dashboard application that requires little to no technical knowledge to setup and utilize from individuals that are a part of the Zyp Art Gallery organization. The long term benefit of this project would be that it would assist members, volunteers, and current/potential stakeholders to assess the social media outreach and human impact of the non-profit organization. In turn, the organization can then make strategic decision that could contribute to its growth in terms of popularity, growth of volunteers, overall expansion of both art-related & charitable endeavors, and donations/funding.
Project Purpose that should address the core problem in terms of the benefits to be received by the project beneficiaries or target group as a direct result of the project	The purpose of this project would be to construct an easily accessible custom dashboard application for the internal use of Zyp Art Gallery members, volunteers, and prospectively current/potential stakeholders (upon the discretion of the Calmar Art Society). This would allow the aforementioned entities to derive insights from informative, aesthetic, and interactive visualizations created from extracted Facebook & Instagram posts, page insights, and demographics data. The organization would be able to understand its social media audience better. Below is a list of some of insights they would be able to discover in a visual manner. ● The number of likes, shares, and comments of specific posts, as well as the organic reach of those posts. ● Overall engagement, total likes, total views, number of followers over a period of time. ● Distribution of page viewers & followers over a period of time in terms age & gender, country, and Canadian city. ● Amount of activity by time range within a 24 hour day over a period of time.
Outputs i.e. results describe the services or products to be delivered to the intended	The following are links to Facebook & Instagram data reference guides that explain the type of data that would be visualized in the form of charts, graphs, and diagrams within the custom dashboard application. The data that could potentially be visualized could expand based on the request of the Analytics & Marketing teams. Facebook: https://github.com/kjeshang/ZypArtGallery-SocialMediaDataExtraction/blob/

beneficiaries	main/ZypFacebook/Facebook%20Data%20Reference%20Guide.pdf Instagram: https://github.com/kjeshang/ZypArtGallery-SocialMediaDataExtraction/blob/main/ZypInstagram/Instagram%20Data%20Reference%20Guide.pdf

Target group & Management and personnel

As aforementioned, the target group overall is the Zyp Art Gallery non-profit organization. There are several different teams within the organization. In particular, the Analytics and Marketing teams would find the custom dashboard application highly useful. By extension, the executive director, Ms. Jaime Rathor, would also use the custom dashboard application as she supervises all volunteers that are part of the teams within the Zyp Art Gallery. I inevitably directly report to her. The Calmar Art Society, which includes the founder & president of Zyp Art Gallery, could potentially use this tool as its members are long term stakeholders of the organization. In the context of this project, below are all of the people I would potentially be reporting to. The table is ordered from left-to-right in terms of descending level of importance of a respective sub-target group. That being said, the table below excludes another target group which is my instructor, Prof. Stephen Chiong, who is also someone I would have to report to through that the stages of this project.

Supervisor (1)	Analytics team (7)	Marketing team (5)	Calmar Art Society (4)
Jaime Rathor (Executive Director/Curator)	• Tanmay (Team Leader) • Colin • Dimeji • Isaac • Larissa • Sandeep • Zeynep	• Dini • Gaurav • Juan David Blanco • Monda • Veronica Q	• Nhung Tran-Davies (President/Founder) • Jaclyn Evaschyshyn (Chair) • Grant Davies (Secretery) • Elizabeth Welsh (Chair)

Monitoring and evaluation

The project would monitored by the executive director of the Zyp Art Gallery along with the rest of the Analytics team in terms of general direction in regards to what type of data they would like to visualize, any supplementary analyses in regards to the data, and requests for new features. The frequency of receiving their approval and feedback would be on a weekly basis during Analytics team meetings on Discord on Thursdays at 5:00PM Pacific time. The meetings are not exactly scripted and do not follow a set agenda at times as there is plethora of other tasks in the pipeline. That being said, during these weekly meetings whenever I discuss or show my work, I

will document the response and feedback for the purpose of documentation and to serve as evidence of monitoring & evaluation. Documentation would occur after prospective project proposal approval of my instructor. As the project grows, so too would the need of receiving feedback. These additional meetings would be with the executive director as she would be my primary source of feedback. In turn, the outcome & response of these meetings would also be documented. I will document these interactions in table format and order by date of occurrence chronologically.

Below is a list of prospective aspects that would be monitored in regards to the custom dashboard application itself.

- Is the application easy to access by those that are part of the Zyp Art Gallery organization?
- Is the application easy to use even for a non-technical person?
- Is the application overall aesthetically pleasing?
- Does the application follow a logical design in regards to navigation?
- Are the visualizations detailed & informative, yet easy on the eyes?
- Do the visualizations have a good degree of interactivity in terms of date range, category, and/or any other identifier?
- (more prospective achievable aspects could be determined as the project grows over time)

Significance of Study

This project would prove significant to Zyp Art Gallery as it would assist it to derive social media insights, in turn be better informed to make strategic business decisions. These business decisions are clinical to the organization to widen its outreach in terms of endeavors to showcase artwork of local artists, help its surrounding community, provide meaningful experiences to its volunteers, etc. Furthermore, the organization would know what more it needs to do to increase its social media presence, as this can encourage current/prospectvie followers to get involved and make donations, in turn, maintaining/improving the financial longevity and positive impact of the non-profit. In simple words, understanding Zyp Art Gallery's social media presence & audience perception helps it to make smart decisions to widen its outreach to make positive impact on more people.

This project is significant to me as I am making a positive contribution to an organization that's intent is to give a "voice to the voiceless" and providing a selfless-service ti help those in need. I also get to apply the knowledge & skills I have gained as Computing & Information Systems student specializing in Data Analytics. Specifically, I get to apply my software development knowledge to create a multi-layer application. I also get to apply my data analytics knowledge to clean, aggregate, filter, analyze, and visualize data to derive meaningful insights.

Proof of Progress

Project Structure

Below is a generic structure of the social media dashboard application. Please note I have ignored including files & folders that are automatically created from simply running the application using Python (e.g., pycache folder) or generated from the IDE that is running the application (e.g., .idea folder when using PyCharm).

Zyp Social Media Dashboard Application Project Directory

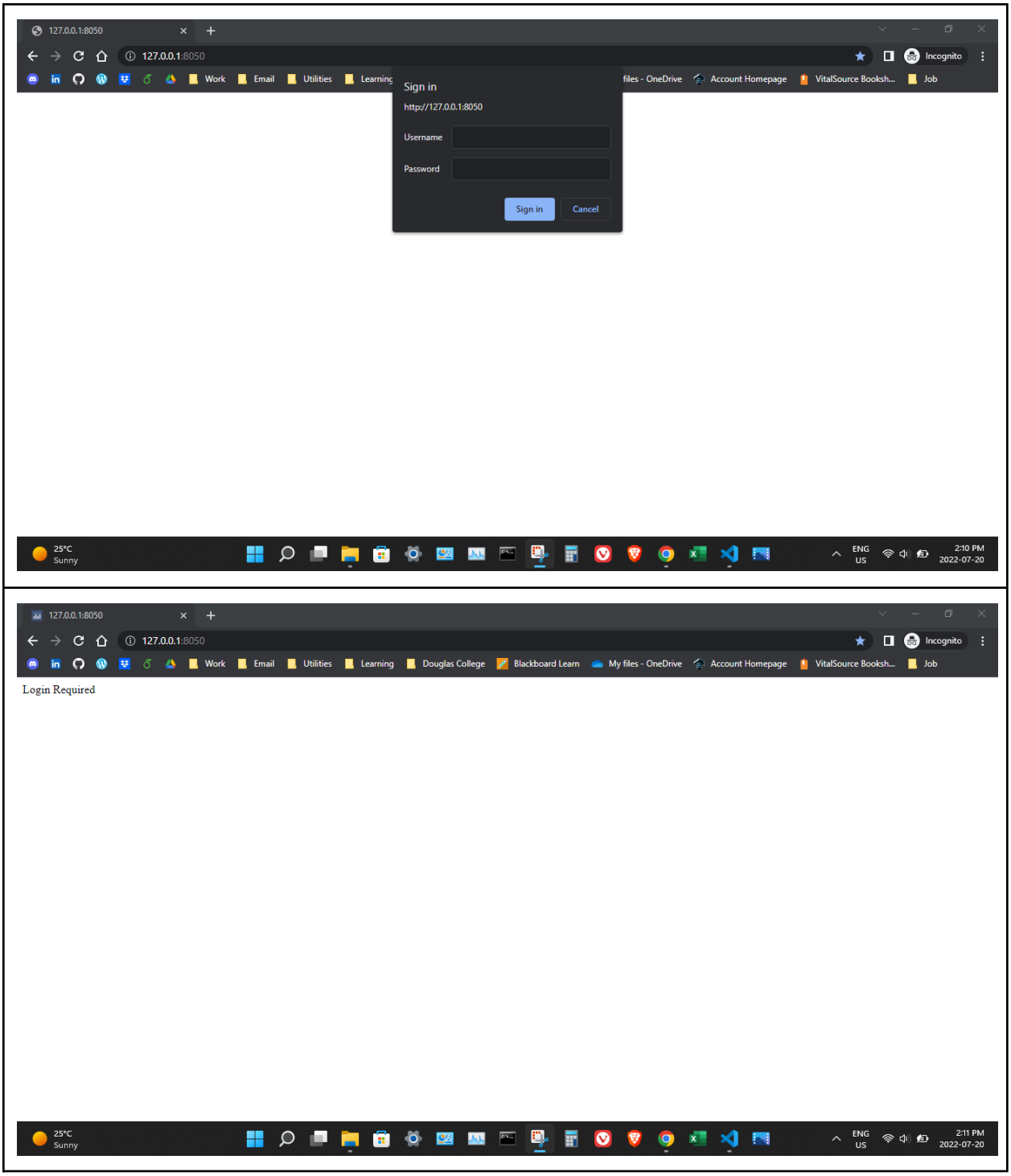
- Apps folder
    - FB_Section_Audience_AgeGender.py
    - FB_Section_Audience_CanadianCity_DailyReach.py
    - FB_Section_Audience_CanadianCity_LifetimeLikes.py
    - FB_Section_Audience_CanadianCity.py (not used anymore – replaced with the two Canadian City scripts above)
    - FB_Section_Audience_Country.py
    - FB_Section_Audience_TimeOfDay.py
    - FB_Section_Page.py
    - FB_Section_Posts.py
    - IG_Section_Audience_AgeGender.py
    - IG_Section_Audience_CanadianCity.py
    - IG_Section_Audience_Country.py
    - IG_Section_Audience_TimeOfDay.py
    - IG_Section_Page.py
    - IG_Section_Posts.py
    - __init__.py
    - home.py
- Assets folder
    - CountryCode-iso_alpha_Final.csv
    - FB_audienceMetrics.py
    - FB_pageMetrics.py
    - FB_postMetrics.py
    - GeoNamesData.csv
    - IG_audienceMetrics.py
    - IG_pageMetrics.py
    - IG_postMetrics.py
    - __init__.py
    - Favicon.ico
    - googleService.py
    - zyp-art-gallery-62e00b2be4ff.json
- Data folder
    - ZypFacebook_Audience-Age&Gender1.csv

    - ZypFacebook_Audience-Age&Gender2.csv
    - ZypFacebook_Audience-CanadianCity1.csv
    - ZypFacebook_Audience-CanadianCity2.csv
    - ZypFacebook_Audience-Country1.csv
    - ZypFacebook_Audience-Country2.csv
    - ZypFacebook_Audience-TimeOfDay.csv
    - ZypFacebook_Insights1.csv
    - ZypFacebook_Insights2.csv
    - ZypFacebook_Insights3.csv
    - ZypFacebook_Posts.csv
    - ZypInstagram_Audience-Age&Gender.csv
    - ZypInstagram_Audience-CanadianCity.csv
    - ZypInstagram_Audience-Country.csv
    - ZypInstagram_Audience-TimeOfDay.csv
    - ZypInstagram_Insights1.csv
    - ZypInstagram_Insights2.csv
    - ZypInstagram_Posts.csv
- Procfile
- App.py
- Index.py
- Requirements.txt
- runtime.txt

Application Screenshots

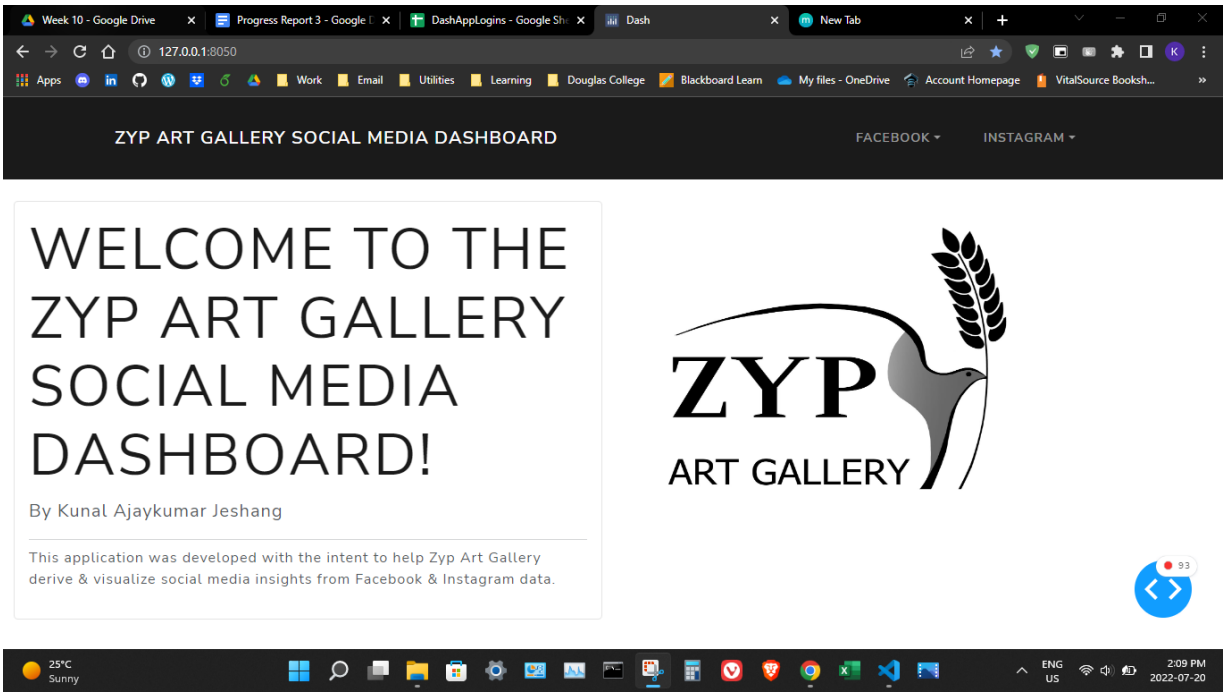
Login & Home Page

Authentication



As can be seen from the above screenshot, upon running the application, the login pop-up would appear. To sign in, one must put in the username & password, and then select the ‘Sign In’ button. The correct credentials are required otherwise it would ask the user again. If the ‘Cancel’ button is selected, then a blank page would be seen with the words “Login Required”.

Home page



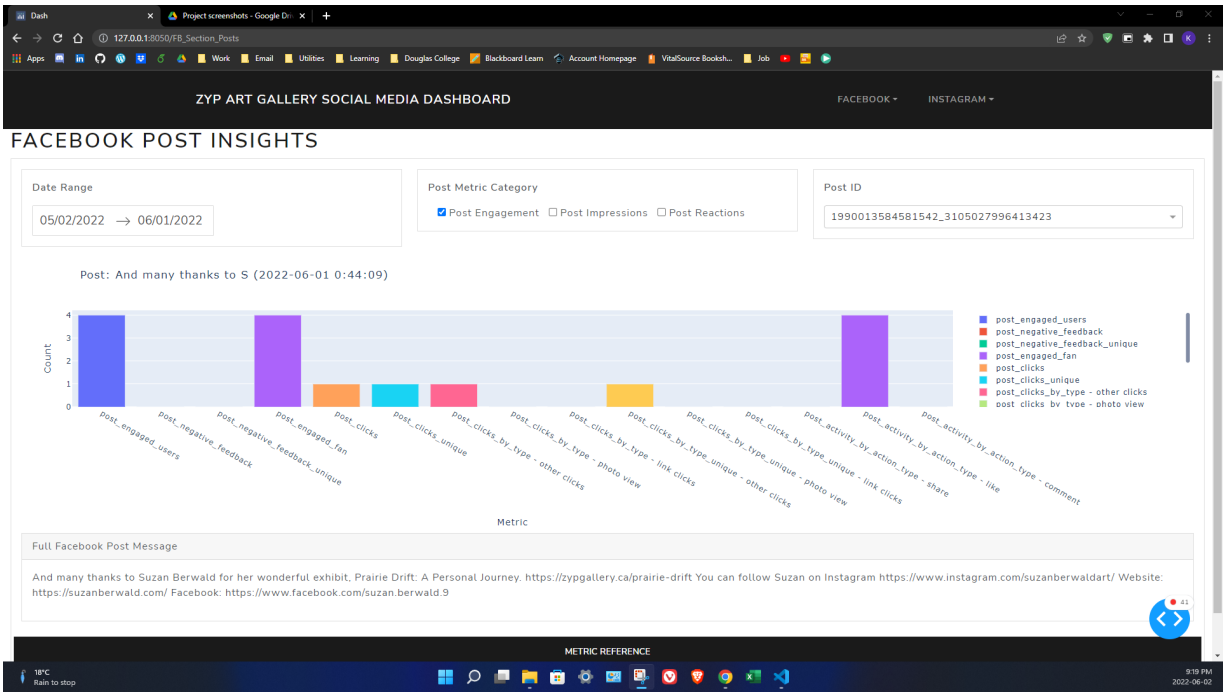
The above screenshot shows the home/landing page of the application. This is the page that is seen when the user manages to successfully authenticate themselves to access the application. In addition notice that the navigation bar appears at the top of the webpage. This navigation bar is standard throughout the rest of of the webpages in subsequent Facebook & Instagram sections. Selecting an option within the Facebook or Instagram dropdown menus will then return the content of the corresponding section of the application. The following is the navigation bar link hierarchy.

- Title of Dashboard Application: Zyp Art Gallery Social Media Dashboard (title on navigation bar and is currently hyperlinked to the index page)
- Facebook section navigation dropdown menu (navlink options available on hover)
    - Posts
    - Page
    - Audience (this is not a link but a header within the navigation bar's dropdown menu)
        - Age & Gender
        - Country
        - Canadian City
        - Time of Day
- Instagram section navigation dropdown menu (navlink options available on hover)
    - Posts
    - Page
    - Audience (this is not a link but a header within the navigation bar's dropdown menu)

    - Age & Gender
    - Country
    - Canadian City
    - Time of Day
- If an invalid link is accessed, or there is any issue acessing the home page and/or the various sections of the application, a page would be displayed with the following message: “404 Page Error! Please choose a link”

Facebook Sections

Facebook Post Insights



As can be seen in the above screenshot shows the Facebook Post Insights webpage. The intent of this page is to communicate the performance of the organization’s Facebook posts.

<u>Interactive elements</u>

The webpage contains two interactive elements that are consistent throughout the other webpages as well: the Date Range datepicker and the Metric Reference button. Metric Reference button toggles a table that appears below it explaining the meaning of the selected metrics that are visible on the visualizations in terms of title and description. The datepicker on this webpage filters the options available in the Post ID dropdown. Specifically, the Post ID refers to a unique identifier for the Facebook posts on the organization’s Facebook page. Thus, the Post ID dropdown contains unique identifiers of Facebook posts that were published within the selected date range from the datepicker. Selecting a specific Post ID from the respective dropdown would change the bar chart visualization to display the metric value counts of the

selected Post ID. The title of the bar chart visualization would also change to display the first 20 character's of the Post ID's message (i.e., actual Facebook post) along with its respective date & time of post. The box showing the full Facebook post, which is below the bar chart, would change to show the entire post message of the selected Post ID. Also notice the Post Metric Category checklist. This checklist groups the various post metrics together based on engagement, impressions, and reactions. The metrics associated with the three categories help to assess the engagement (i.e., overall interactivity with the post), impressions (i.e., reach), and reactions (e.g., like) of the selected post. Selecting all or non of the options in the Post Metric Category checklist would also change the metrics visible in the table that is toggled by the Metric Reference button.

Data

This webpage uses the "ZypFacebook_Posts" Google Sheets document that is saved the Organization's Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
id	Identification number of the facebook
message	The actual post made to facebook
created_time	Date & time of facebook post
date	Date of facebook post (extracted from created_time)
time	Time of facebook post (extracted from created_time)
post	The first 20 characters of the actual facebook post (extracted from message)

Post Engagement:

Metric	***Title & Description***
post_engaged_users	*Lifetime Engaged Users* Lifetime: The number of unique people who engaged in certain ways with your Page post, for example by commenting on, liking, sharing, or clicking upon particular elements of the post. (Unique Users)
post_negative_feedback	*Lifetime Negative Feedback* Lifetime: The number of times people have given negative feedback to your post. (Total Count)

post_negative_feedback_unique	*Lifetime Negative Feedback from Users* Lifetime: The number of people who have given negative feedback to your post. (Unique Users)
post_engaged_fan	*Lifetime People who have liked your Page and engaged with your post* Lifetime: The number of people who have liked your Page and clicked anywhere in your posts. (Unique Users)
post_clicks	*Lifetime Matched Audience Targeting Consumptions on Post* Lifetime: The number of clicks anywhere in your post on News Feed from the user that matched the audience targeting on it. (Total Count)
post_clicks_unique	*Lifetime Matched Audience Targeting Consumers on Post* Lifetime: The number of people who matched the audience targeting that clicked anywhere in your post on News Feed. (Unique Users)
post_clicks_by_type - other clicks post_clicks_by_type - photo view post_clicks_by_type - link clicks	*Lifetime Matched Audience Targeting Consumptions by Type* Consumption Type: other clicks, photo view, link clicks Lifetime: The number of clicks anywhere in the post on News Feed from users that matched the audience targeting on the post, by type. (Total Count)
post_clicks_by_type_unique - other clicks post_clicks_by_type_unique - photo view post_clicks_by_type_unique - link clicks	*Lifetime Post Audience Targeting Unique Consumptions by Type* Consumption Type: other clicks, photo view, link clicks Lifetime: The number of people who matched the audience targeting on the post that clicked anywhere in the post on News Feed, by type. (Unique Users)

post_activity_by_action_type - share post_activity_by_action_type - like post_activity_by_action_type - comment	*Lifetime Post Stories by action type* Action Type: share, like, comment Lifetime: The number of stories created about your Page post, by action type. (Total Count)

Post Impressions:

Metric	***Title & Description***
post_impressions	*Lifetime Post Total Impressions* Lifetime: The number of times your Page's post entered a person's screen. Posts include statuses, photos, links, videos and more. (Total Count)
post_impressions_unique	*Lifetime Post Total Reach* Lifetime: The number of people who had your Page's post enter their screen. Posts include statuses, photos, links, videos and more. (Unique Users)
post_impressions_paid	*Lifetime Post Paid Impressions* Lifetime: The number of times your Page's post entered a person's screen through paid distribution such as an ad. (Total Count)
post_impressions_paid_unique	*Lifetime Post Paid Reach* Lifetime: The number of people who had your Page's post enter their screen through paid distribution such as an ad. (Unique Users)
post_impressions_fan	*Lifetime Post Impressions by people who have liked your Page* Lifetime: The number of impressions of your Page post to people who have liked your Page. (Total Count)
post_impressions_fan_unique	*Lifetime Post reach by people who like your Page* Lifetime: The number of people who saw your Page post because they've liked your Page (Unique Users)

post_impressions_fan_paid	*Lifetime Post Paid Impressions by people who have liked your Page* Lifetime: The number of paid impressions of your Page post to people who have liked your Page. (Total Count)
post_impressions_fan_paid_unique	*Lifetime Paid reach of a post by people who like your Page* Lifetime: The number of people who like your Page and who saw your Page post in an ad or sponsored story. (Unique Users)
post_impressions_organic	*Lifetime Post Organic Impressions* Lifetime: The number of times your Page's posts entered a person's screen through unpaid distribution. (Total Count)
post_impressions_organic_unique	*Lifetime Post organic reach* Lifetime: The number of people who had your Page's post enter their screen through unpaid distribution. (Unique Users)
post_impressions_viral	*Lifetime Post Viral Impressions* Lifetime: The number of times your Page's post entered a person's screen with social information attached. Social information displays when a person's friend interacted with you Page or post. This includes when someone's friend likes or follows your Page, engages with a post, shares a photo of your Page and checks into your Page. (Total Count)
post_impressions_viral_unique	*Lifetime Post viral reach* Lifetime: The number of people who had your Page's post enter their screen with social information attached. As a form of organic distribution, social information displays when a person's friend interacted with you Page or post. This includes when someone's friend likes or follows your Page, engages with a post, shares a photo of your Page and checks into your Page. (Unique Users)

post_impressions_nonviral	*Lifetime Post Nonviral Impressions* Lifetime: The number of times your Page's post entered a person's screen. This does not include content created about your Page with social information attached. Social information displays when a person's friend interacted with you Page or post. This includes when someone's friend likes or follows your Page, engages with a post, shares a photo of your Page and checks into your Page. (Total Count)
post_impressions_nonviral_unique	*Lifetime Post Nonviral Reach* Lifetime: The number of people who had your Page's post enter their screen. This does not include content created about your Page with social information attached. As a form of organic distribution, social information displays when a person's friend interacted with you Page or post. This includes when someone's friend likes or follows your Page, engages with a post, shares a photo of your Page and checks into your Page. (Unique Users)
post_impressions_by_story_type	*Lifetime Post Viral Impressions by story type* Lifetime: The number of times people saw this post via stories published by their friends. (Total Count)
post_impressions_by_story_type_unique	*Lifetime Post viral reach by story type* Lifetime: The number of people who saw your Page post in a story from a friend, by story type. (Unique Users)

Post Reactions:

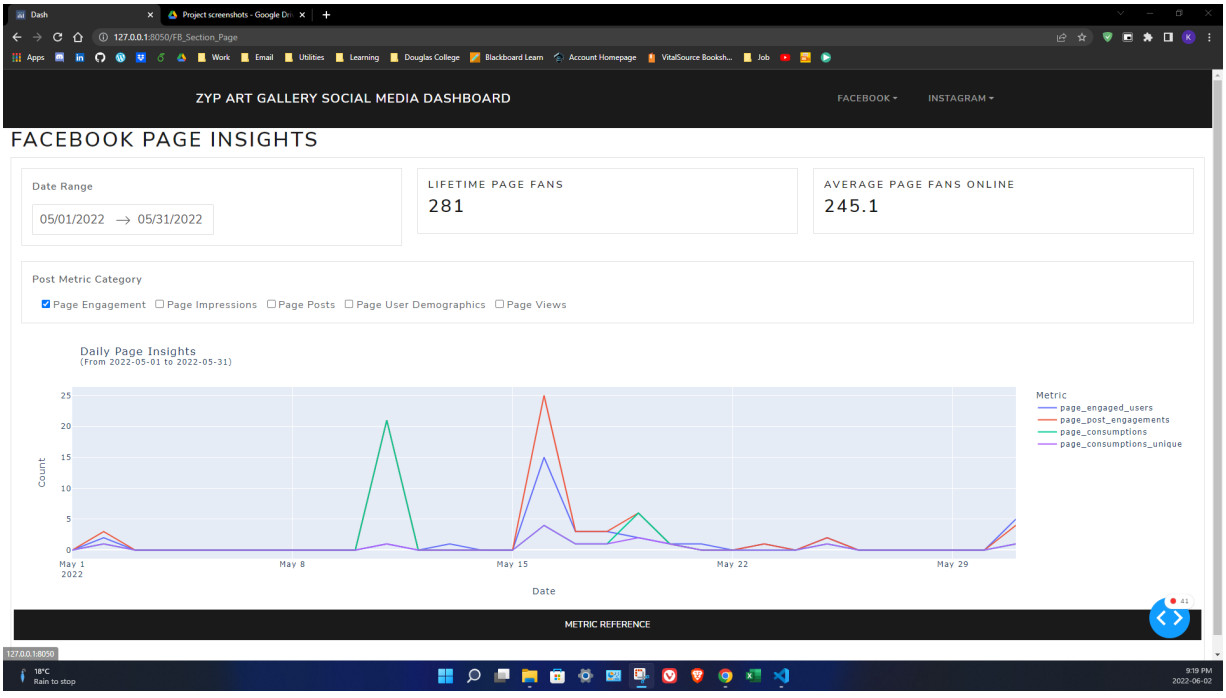
Metric	***Title & Description***
post_reactions_like_total	*Lifetime Total Like Reactions of a post* Lifetime: Total like reactions of a post.
post_reactions_love_total	*Lifetime Total Love Reactions of a post* Lifetime: Total love reactions of a post.

post_reactions_wow_total	*Lifetime Total wow Reactions of a post* Lifetime: Total wow Reactions of a post.
post_reactions_haha_total	*Lifetime Total haha Reactions of a post* Lifetime: Total haha reactions of a post.
post_reactions_sorry_total	*Lifetime Total sad Reactions of a post* Lifetime: Total sad reactions of a post.
post_reactions_anger_total	*Lifetime Total anger Reactions of a post* Lifetime: Total anger reactions of a post.

Visualization

As most of the insight related metrics are of period ‘Lifetime’, no aggregations or further calculations were done to create the visualization. Thus, I am taking the metrics at face value. Thus, I believed to best communicate the performance of the Facebook post would be to create a bar chart comparing the metric values to each other. I also thought it would be helpful for a potential user to actually read the entire Facebook post in full form, which is available below the bar chart. This way the user could gauge their own deduction of the metric values more intuitively and organically.

Facebook Page Insights



As can be seen in the above screenshot shows the Facebook Page Insights webpage. The intent of this page is to communicate the performance of the organization's Facebook page overall.

Interactive elements

As aforementioned, the date range datepicker is consistent element throughout the webpages of the application. On this webpage, it filters the page insight metric values that are within the respective date range on the line chart as per day. The date range that is selected is visible on the line chart title. The line chart's x-axis starts at the start date and ends at the end date of the date range shown in the datepicker. The datepicker alters the lifetime page fans tile number based on the end date selected. The datepicker also alters average page fans online tile number based on both the start & end date selected. By default, all the datepickers show a default date range of the last 30 days. The options in the page metric category checklist groups page insight metrics into specific categories; page engagement, impressions, page posts, user demographics, and page views.

Data

This webpage uses the "ZypFacebook_Insights1", "ZypFacebook_Insights2", and "ZypFacebook_Insights3" Google Sheets document that are saved on the Organization's Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

The following refers to information visualized on the lifetime page fans tile.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans	*Lifetime Total Likes* Lifetime: The total number of people who have liked your Page. (Unique Users)

The following refers to information visualized on the average page fans online tile.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_online_per_day	*Daily liked and Online by Day*

	Daily: The number of people who liked your Page and who were online on the specified day. (Unique Users)

The following information is visualized on the line chart.

Field	***Description***
end_time	Date

Page Engagment:

Metric	***Title & Description***
page_engaged_users	*Daily Page Engaged Users* Daily: The number of people who engaged with your Page. Engagement includes any click or story created. (Unique Users)
page_post_engagements	*Daily Post Engagements* Daily: The number of times people have engaged with your posts through like, comments and shares and more.
page_consumptions	*Daily Page Consumptions* Daily: The number of clicks on any of your content. Stories generated without clicks on page content (e.g., liking the page in Timeline) are not included. (Total Count)
page_consumptions_unique	*Daily Total Consumers* Daily: The number of people who clicked on any of your content. Stories that are created without clicking on Page content (ex, liking the Page from timeline) are not included. (Unique Users)

Page Impressions:

Metric	***Title & Description***
page_impressions	*Daily Total Impressions* Daily: The number of times any content from your Page or about your Page entered a person's screen. This includes posts, stories, ads, as well other content or information on your Page. (Total Count)
page_impressions_unique	*Daily Total Reach*

	Daily: The number of people who had any content from your Page or about your Page enter their screen. This includes posts, check-ins, ads, social information from people who interact with your Page and more. (Unique Users)

Page Posts Impressions (i.e., Page Posts):

Metric	***Title & Description***
page_posts_impressions	*Daily Total Impressions of your posts* Daily: The number of times your Page's posts entered a person's screen. Posts include statuses, photos, links, videos and more. (Total Count)
page_posts_impressions_unique	*Daily Reach Of Page Posts* Daily: The number of people who had any of your Page's posts enter their screen. Posts include statuses, photos, links, videos and more. (Unique Users)

Page User Demographics:

Metric	***Title & Description***
page_fan_adds	*Daily New Likes* Daily: The number of new people who have liked your Page (Total Count)
page_fan_adds_unique	*Daily New Likes Unique* Daily: The number of new people who have liked your Page (Unique Users)
page_fan_removes	*Daily Unlikes* Daily: The number of Unlikes of your Page (Total Count)
page_fan_removes_unique	*Daily Unlikes Unique* Daily: The number of Unlikes of your Page (Unique Users)

Page Views:

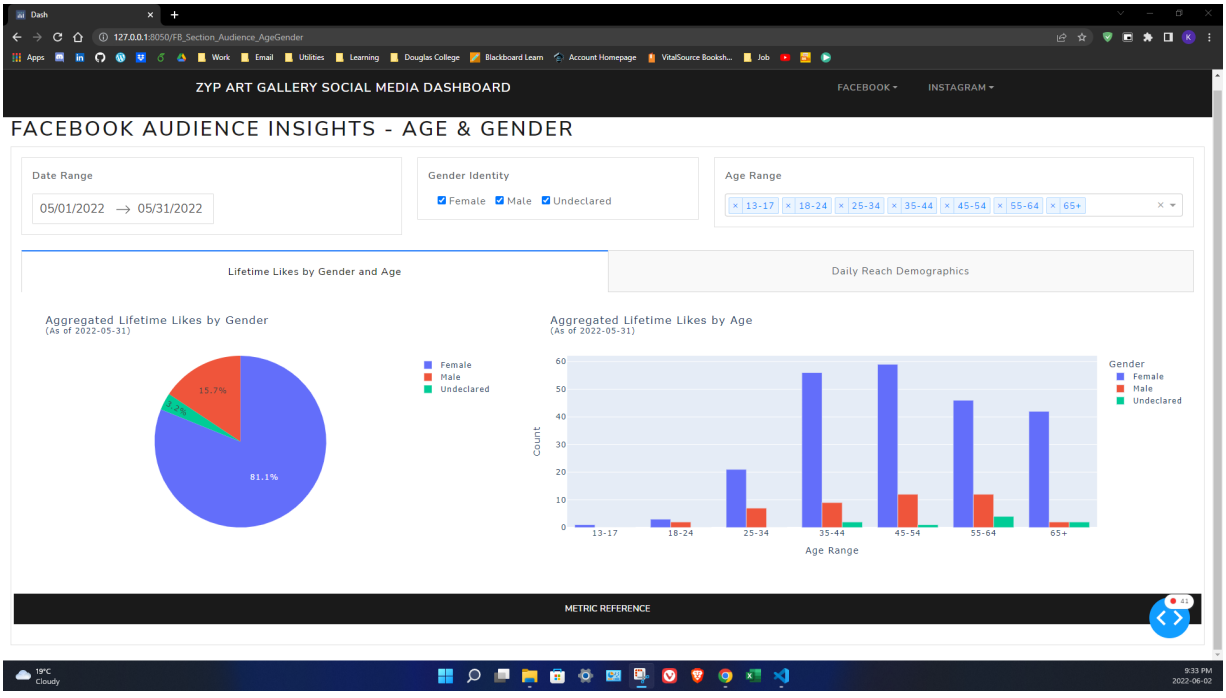
Metric	***Title & Description***
page_views_total	*Daily Total views count per Page*

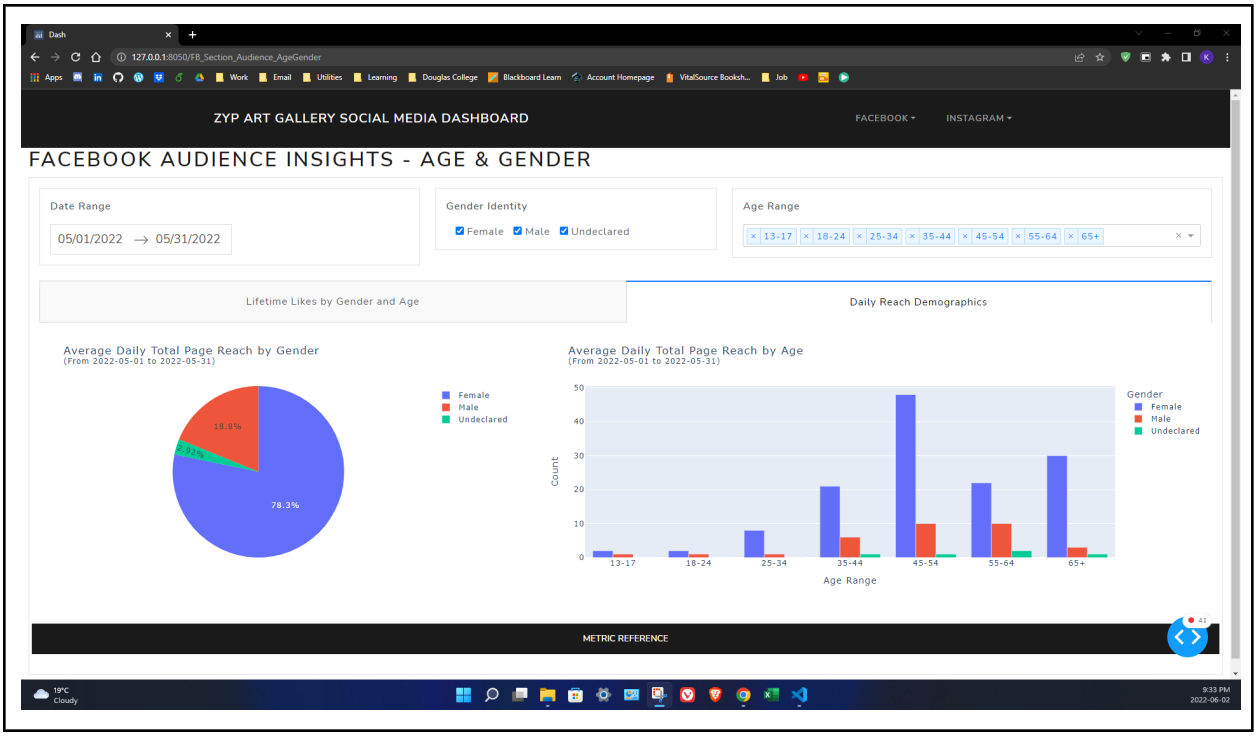
	Daily: Total views count per Page
page_views_logged_in_total	*Daily Total logged-in views count per Page* Daily: Total logged-in views count per Page
page_views_logged_in_unique	*Daily Total logged-in views count per Page* Daily: Total logged-in views count per Page (Unique Users)

Visualization

As there are many page insight metrics that are of the period “Daily”, I felt it would be interesting to track these metrics over a period of time (i.e. per day) to assess the performance fluctuations of the Facebook page. Specifically, the “ZypFacebook_Insights1” Google Sheets data file was used to create the line chart. The lifetime page fans tile utilizes the “ZypFacebook_Insights3” Google Sheets Data file. It is of period “Lifetime”, so it simply takes the metric value that is as of the end date selected on the date range datepicker. Furthermore, there is no need to perform aggregations or further calculations. However, as the average page fans online is of period “Daily”, it would make sense to perform an average aggregation of metric values from the start and end date displayed on the date range datepicker. The “ZypFacebook_Insights2” Google Sheets data file is used to calculate the number in the average page fans online tile.

Facebook Audience Insights - Age & Gender





As can be seen in the above screenshot shows the Facebook Audience Insights - Age & Gender webpage. The overall intent of this webpage is to assess the performance of the Facebook page by age and gender.

Interactive elements

There are two tabs. The intent of the first tab is to communicate the likes of the Facebook page by Age & Gender. The metric used in this tab is of period “Lifetime”. Thus, the visualizations change based on the end date selected on the date range datepicker. The intent of the second tab it to communicate the reach of the Facebook page by Age & Gender. The metric used in this tab is of period “Daily”. Thus, the visualizations change based on the start & date selected on the date range picker, and calculates the average of the metric value based on date range. The titles of the visualizations in both tabs change based on the selections made in the datepicker. The gender identity checklist and age range dropdowns also filter and adjust the visualizations accordingly. If a gender option or age range option is de-selected, the visualizations would not display them, and associate them with any calculation or aggregation leading up to rendering the charts. Although, by default all gender and age options are selected.

Data

This webpage uses the “ZypFacebook_Audience-Age&Gender1” and “ZypFacebook_Audience-Age&Gender2” Google Sheets documents that are saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

The following refers to information visualized on the ‘Lifetime Likes by Gender and Age’ tab.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_gender_age [Gender abbreviation].[Age interval] E.g. F.13-17 (Female aged 13 to 17) ● F = Female ● M = Male ● U = Unknown	*Lifetime Likes by Gender and Age* Lifetime: Aggregated demographic data about the people who like your Page based on the age and gender information they provide in their user profiles. (Unique Users)

The following referes to information visualized on the ‘Daily Reach Demographics’ tab.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_impressions_by_age_gender_unique [Gender abbreviation].[Age interval] => E.g. F.13-17 (Female aged 13 to 17) ● F = Female ● M = Male ● U = Unknown	*Daily Reach Demographics* Daily: Total Page Reach by age and gender. (Unique Users)

Visualization

Regardless of which tab is in view, there are two charts. The pie chart shows the percentage proportion of the metric by gender identity. The side-by-side bar chart compares the metric values by gender across a set of age ranges. The ‘Lifetime Likes by Gender & Age’ tab uses the “ZypFacebook_Audience-Age&Gender1” Google Sheet data file, whilst the ‘Daily Reach Demographics’ tab uses the “ZypFacebook_Audience-Age&Gender2” Google Sheet data file.

Facebook Audience Insights - Country

The above screenshots show the Facebook Audience Insights - Country webpage. As aforementioned in the 'Progress Report 1' section of this report, the intent is visualize the aggregated lifetime likes and average daily reach of the organization's Facebook page by country.

<u>Interactive elements</u>

There are two tabs. The intent of the first tab is to communicate the likes of the Facebook page by Country. The metric used in this tab is of period "Lifetime". Thus, the visualizations change based on the end date selected on the date range datepicker. The intent of the second tab is to communicate the reach of the Facebook page by Country. The metric used in this tab is of period "Daily". Thus, the visualizations change based on the start & date selected on the date

range picker, and calculates the average of the metric value based on date range. The titles of the visualizations in both tabs change based on the selections made in the datepicker.

This webpage also has a region scope radiobutton list. By default, the option selected is “World”. If another region scope option is selected, the choropeth would change to show only selected region and the bar chart would adjust to show the top 10 countries based on the metric that are within the selected region. For example, if “North America” is selected, then the chorpeth would display only North America in view and zoomed in (all other region would not be in view), and the bar chart would display top 10 countries based on the metric within North America.

Data

This webpage uses the “ZypFacebook_Audience-Country1” and “ZypFacebook_Audience-Country2” Google Sheets documents that are saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

The following refers to information visualized on the ‘Lifetime Likes by Country’ tab.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_country Country name => E.g. South Africa	*Lifetime Likes by Country* Lifetime: Aggregated Facebook location data, sorted by country (top 50), about the people who like your Page. (Unique Users)

The following referes to information visualized on the ‘Daily Reach by Country’ tab.

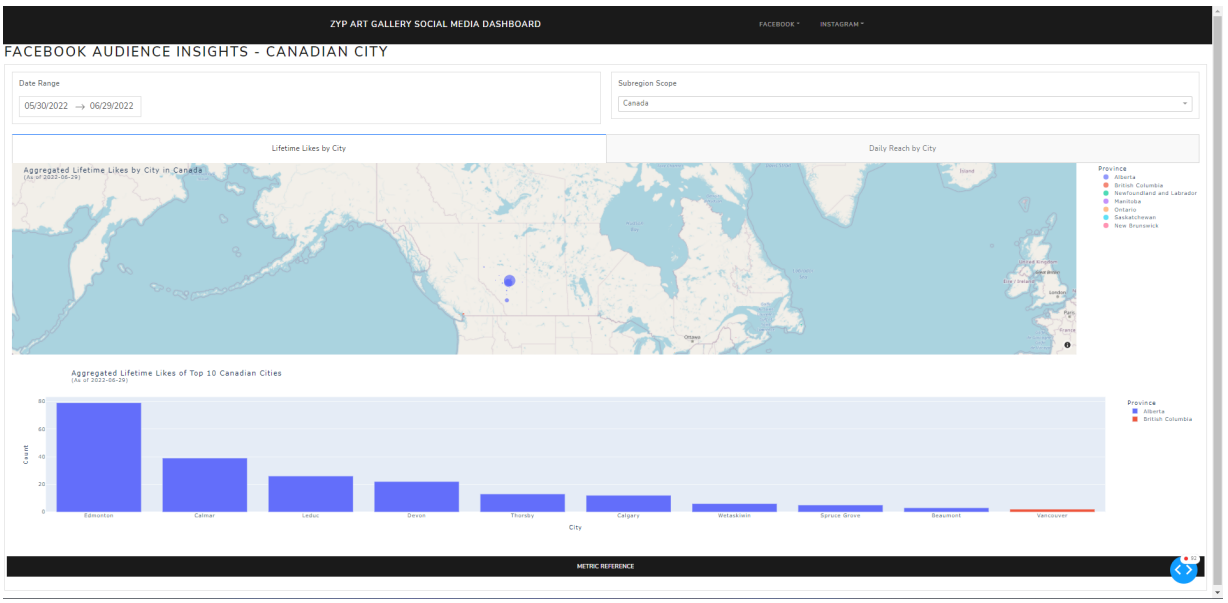
Field	***Description***
end_time	Date
Metric	***Title & Description***
page_impressions_by_country_unique Country name => E.g. South Africa	*Daily Reach by Country* Daily: Total Page Reach by user country. (Unique Users)

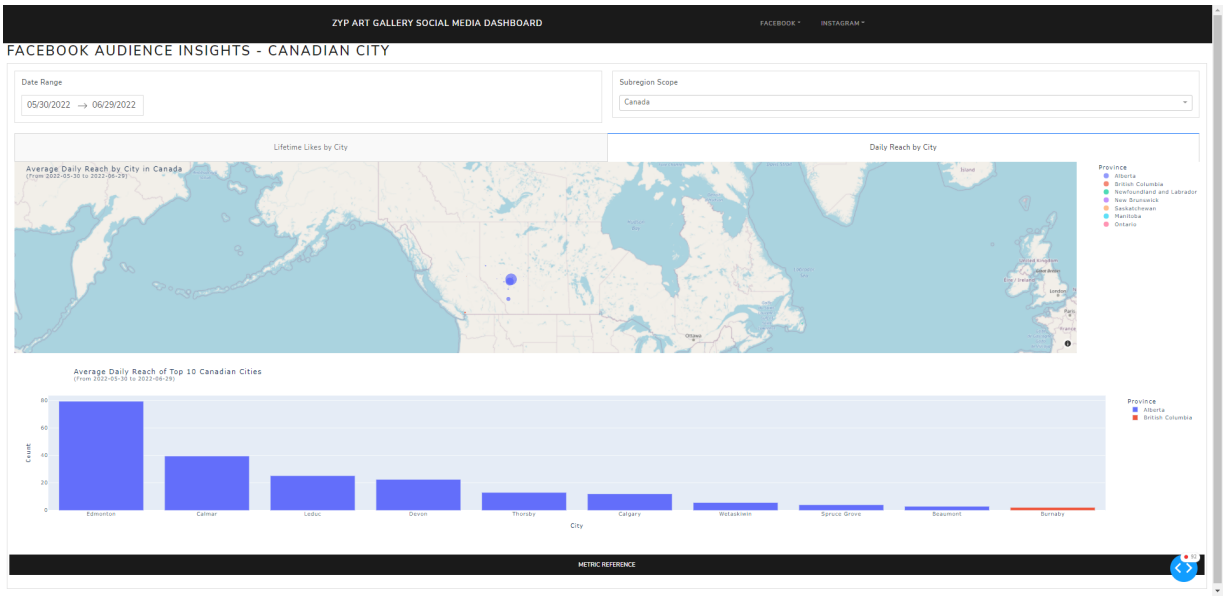
Visualization

Regardless of which tab is in view, there are two charts. The bar chart shows the top countries regarding the metric value. The choropleth bar chart shows the metric value of all countries in

the form of a map. The ‘Lifetime Likes by Country’ tab uses the “ZypFacebook_Audience-Country1” Google Sheet data file, whilst the ‘Daily Reach by Country’ tab uses the “ZypFacebook_Audience-Country2” Google Sheet data file.

Facebook Audience Insights - Canadian City





The above screenshots show the Facebook Audience Insights - Canadian City webpage. The intent is visualize the aggregated lifetime likes and average daily reach of the organization’s Facebook page by Canadian city.

<u>Interactive elements</u>

There are two tabs. The intent of the first tab is to communicate the likes of the Facebook page by Canadian city. The metric used in this tab is of period “Lifetime”. Thus, the visualizations change based on the end date selected on the date range datepicker. The intent of the second tab is to communicate the reach of the Facebook page by Canadian city. The metric used in this tab is of period “Daily”. Thus, the visualizations change based on the start & date selected on the daterange picker, and calculates the average of the metric value based on date range. The titles of the visualizations in both tabs change based on the selections made in the datepicker. The subregion scope dropdown helps to filter the visualizations by province. However, there is also an option in the dropdown called “Canada” which is set to default. If another option selected, which is the name of a province in Canada, the visualizations would adjust to only reflect the selected province in the subregion scope dropdown.

Data

This webpage uses the “ZypFacebook_Audience-CanadianCity1” and “ZypFacebook_Audience-CanadianCity2” Google Sheets documents that are saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

The following refers to information visualized on the ‘Lifetime Likes by City’ tab.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_city City name, Province abbreviation, Country => E.g. Calgary, AB, Canada	*Lifetime Likes by City* Lifetime: Aggregated Facebook location data, sorted by city (top 50), about the people who like your Page. (Unique Users)

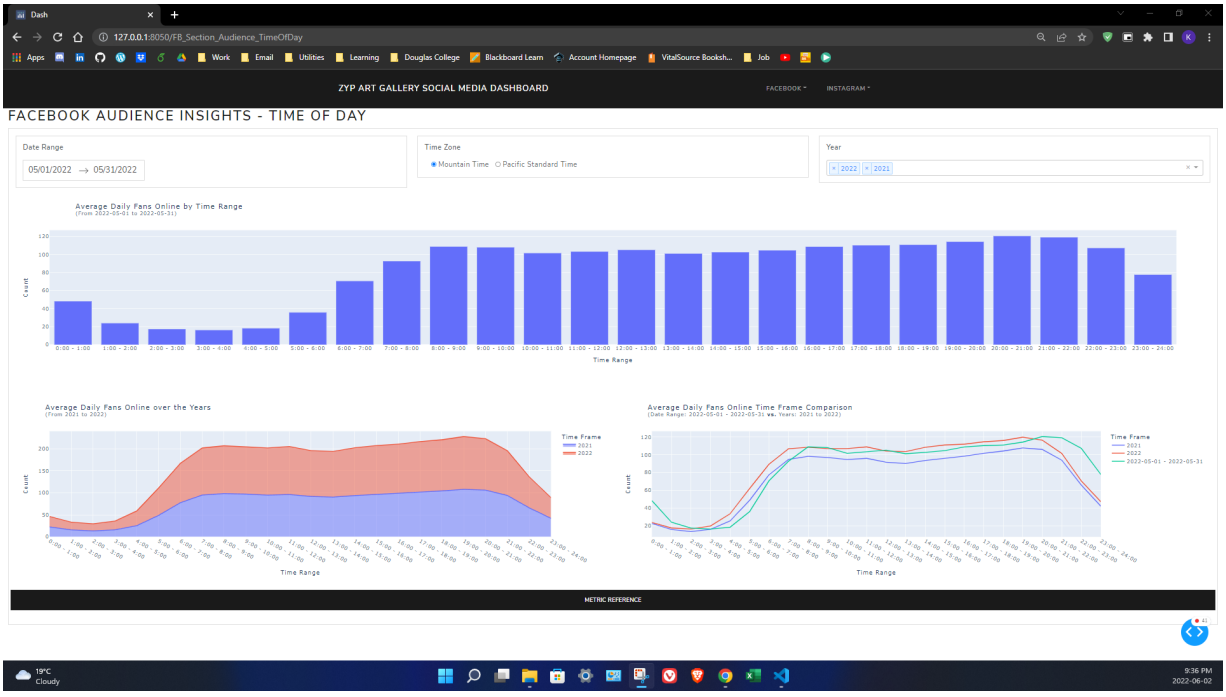
The following referes to information visualized on the ‘Daily Reach by City’ tab.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_impressions_by_city_unique City name, Province abbreviation, Country => E.g. Calgary, AB, Canada	*Daily Reach by City* Daily: Total Page Reach by user city. (Unique Users)

Visualization

Regardless of which tab is in view, there are two charts. The bar chart shows the top Canadian cities regarding the metric value. The bubble map chart shows the metric value of all Canadian cities. If there is selected option in the subregion scope that is other than “Canada”, then the bubble map chart would display bubbles referring to the metric value of Canadian cities within the selected subregion (i.e., Province). The bar chart shows the top 10 Canadian cities based on the metric value. If there is selected option in the subregion scope that is other than “Canada”, then the bar chart chart would display bars referring to the metric value of Canadian cities within the selected subregion (i.e., Province). The ‘Lifetime Likes by Canadian City’ tab uses the “ZypFacebook_Audience-CanadianCity1” Google Sheet data file, whilst the ‘Daily Reach by CanadianCity’ tab uses the “ZypFacebook_Audience-CanadianCity2” Google Sheet data file.

Facebook Audience Insights - Time of Day



As can be seen in the above screenshot shows the Facebook Audience Insights - Time of Day webpage. The overall intent of this webpage is to assess the performance of the Facebook page by time of day.

Interactive elements

The date range datepicker changes the bar chart visualization and line chart visualization. This is specifically so as the aforementioned visualizations calculate the average of the page fans online based on the start and end date of the datepicker. The radio button selector showing the time zones changes all visualizations on the page. The Facebook data is extracted as per Pacific Standard Time by default. Yet as Zyp Art Gallery is an organization based in Alberta that follows Mountain Time, I decided to provide an option to a user to toggle between both time

zones and Mountain Time is set to default. Thus the average page fans online correspond to Mountain Time time ranges if the Mountain Time option is selected. The year dropdown alters the area chart visualization and line chart visualization by including areas and lines respectively that indicate average page fans online per time range.

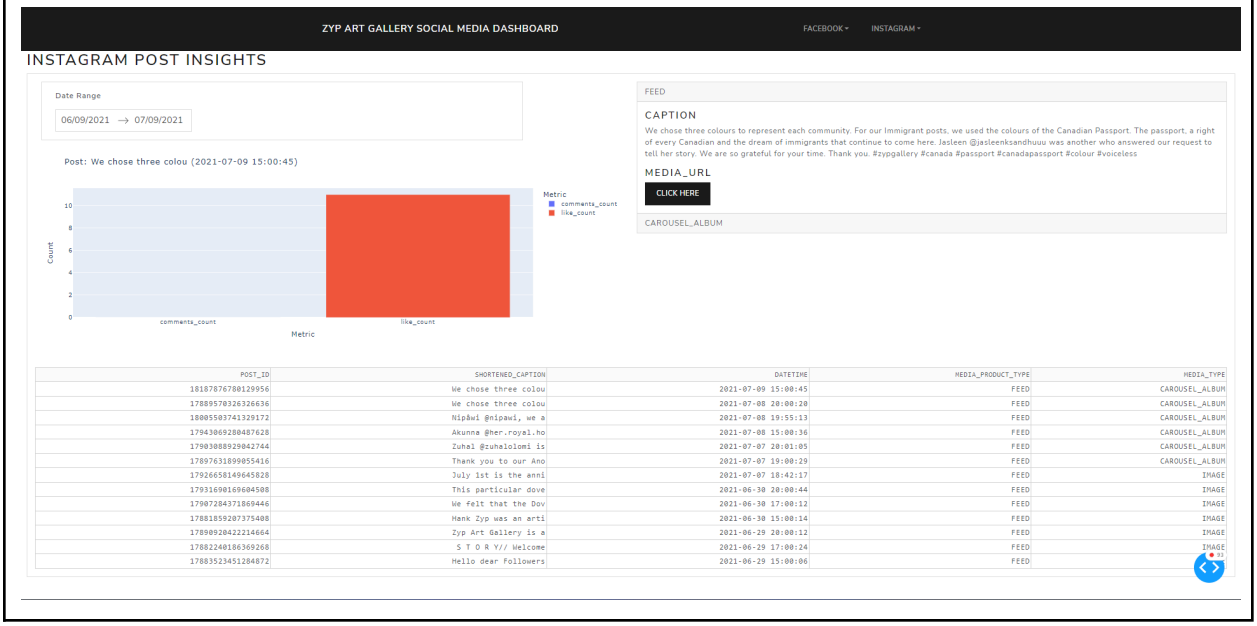
Data

This webpage uses the “ZypFacebook_Audience-TimeOfDay” Google Sheet document that are saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_online Time range in the 24 hour format => E.g. 14:00 - 15:00	*Daily Liked and Online* Daily: The number of people who liked your Page and when they are online in PST/PDT (Unique Users)

Instagram Sections

Instagram Post Insights



The above screenshot shows the Instagram Post Insights webpage. The intent of this page is to communicate the performance of the organization’s Instagram posts.

Interactive elements & Visualization

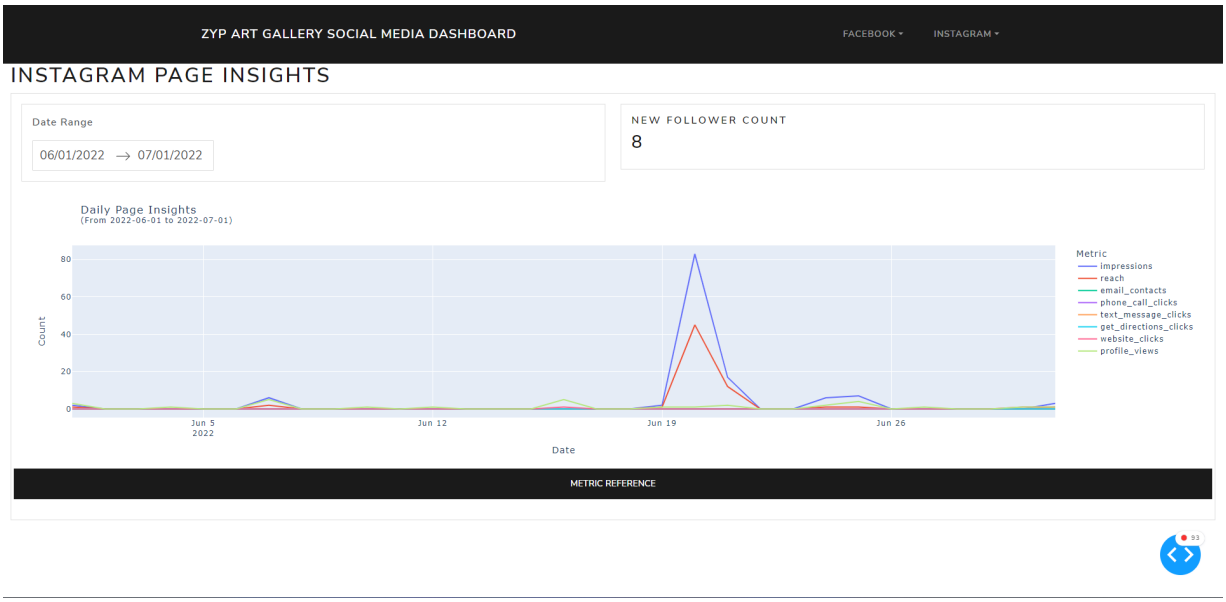
The daterange filter on the top-left side of the triggers the display of the table of the bottom of the page. Specifically, the table displays data of the Instagram posts that within the dates selected on the daterange filter. Upon selection of a row on the table, the Instagram post's information referring to that particular row would be displayed at the top of the page. This would consist of the comment count & like count of the Instagram post communicated in the form of a bar chart, as well as the full Instagram caption, media url, media product type, and media type in the form of a card. The media url is hyperlinked to the button, although upon clicking it a webpage opens saying that the "URL signature has expired". A reason why the card containing the full Instagram caption is required is because the table does not render appropriately when there are column values with a very long character length. Thus, only key information is included in the table which consists of the post ID, first 20 characters of the caption, datetime of post, media product type, and media type.

Data

This webpage uses the "ZypInstagram_Posts" Google Sheets document that is saved the Organization's Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
timestamp	Timestamp of the Instagram post
id	Instagram post identification number
caption	The actual post made to Instagram
media_url	Link to the actual Instagram post
comments_count	Total count of comments received by the Instagram post
like_count	The number of likes received by the Instagram post
media_product_type	The product type of the Instagram post
media_type	The type of post which can be either an image, video, or carousal album
datetime	Date & time of the Instagram post (extracted from timestamp)
date	Date of the Instagram post (extracted from timestamp)
time	Time of the Instagram post (extracted from timestamp)
shortened_caption	The first 20 characters of the actual Instagram post (extracted from caption)

Instagram Page Insights



The above screenshot shows the Instagram Page Insights webpage. The intent of this page is to communicate the performance of the organization's Instagram page overall.

Interactive elements & Visualization

The line chart on this page is used to communicate the Instagram page performance via various metrics over a period of time. The daterange filter adjusts the line chart to display metric values within the dates selected in the daterange filter. The new follower count tile shows the sum total of new followers of the Instagram page within the dates selected in the daterange filter.

Data

This webpage uses the "ZypInstagram_Insights1" (used by the line chart) and "ZypInstagram_Insights2" (used by the new follower count tile) Google Sheets documents that are saved on the Organization's Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

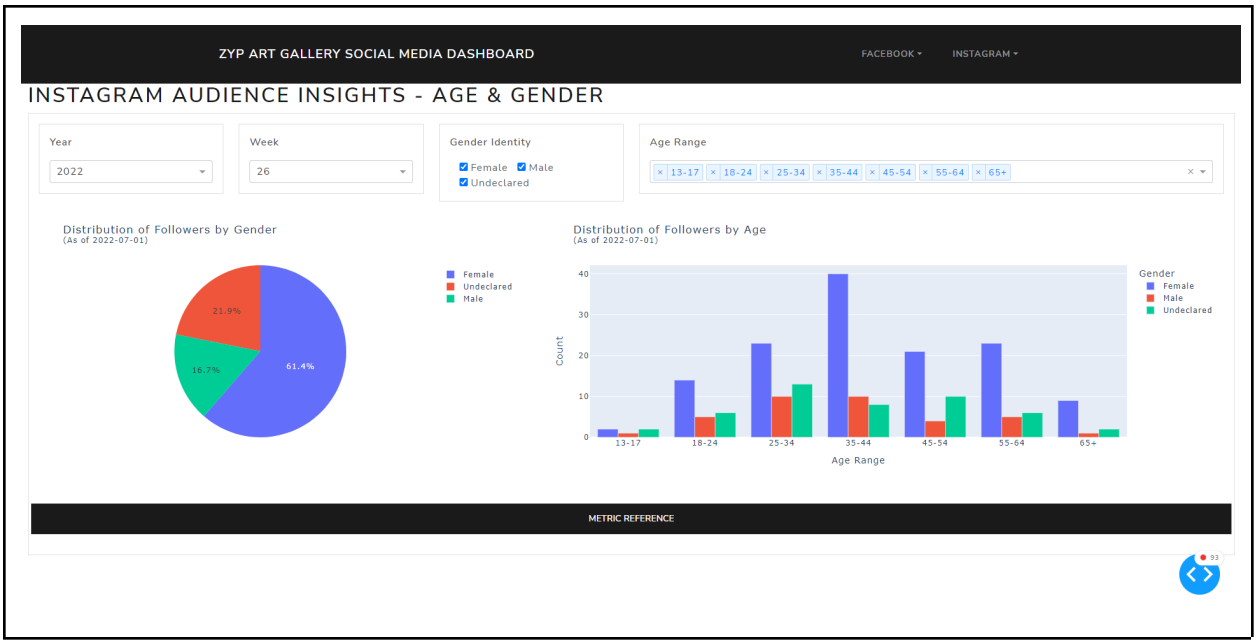
The following refers to information visualized on the new follower count tile.

Field	***Description***
end_time	Date
Metric	***Title & Description***
follower_count	*Follower Count* Total number of unique accounts following this profile

The following refers to information visualized on the line chart.

Field	***Description***
end_time	Date
Metric	***Title & Description***
impressions	*Impressions* Total number of times the Business Account's media objects have been viewed
reach	*Reach* Total number of times the Business Account's media objects have been uniquely viewed
email_contacts	*Email Contacts* Total number of taps on the email link in this profile
phone_call_clicks	*Phone Call Clicks* Total number of taps on the call link in this profile
text_message_clicks	*Text Message Clicks* Total number of taps on the text message link in this profile
get_directions_clicks	*Get Directions Clicks* Total number of taps on the directions link in this profile
website_clicks	*Website Clicks* Total number of taps on the website link in this profile
profile_views	*Profile Views* Total number of users who have viewed the Business Account's profile within the specified period

Instagram Audience Insights - Age & Gender



The above screenshot shows the Instagram Audience Insights - Age & Gender webpage. The overall intent of this webpage is to assess the performance of the Instagram page by age and gender.

Interactive elements & Visualization

As touched upon earlier in this section of the report, all audience related metrics are of period Lifetime per week. Thus, the daterange filter is replaced by the year & week dropdown filter. It functions in the same way as the daterange filter. The gender identity and age range filters function in a similar way as that of the corresponding Facebook section. Changing the selected option in the year and/or week dropdown filters would change the pie chart and side-by-side bar chart accordingly to reflect the metric value as of the selected year & week. The gender identity and age range filters would also further adjust the visualizations accordingly. By default the current year and prior to the current week is selected.

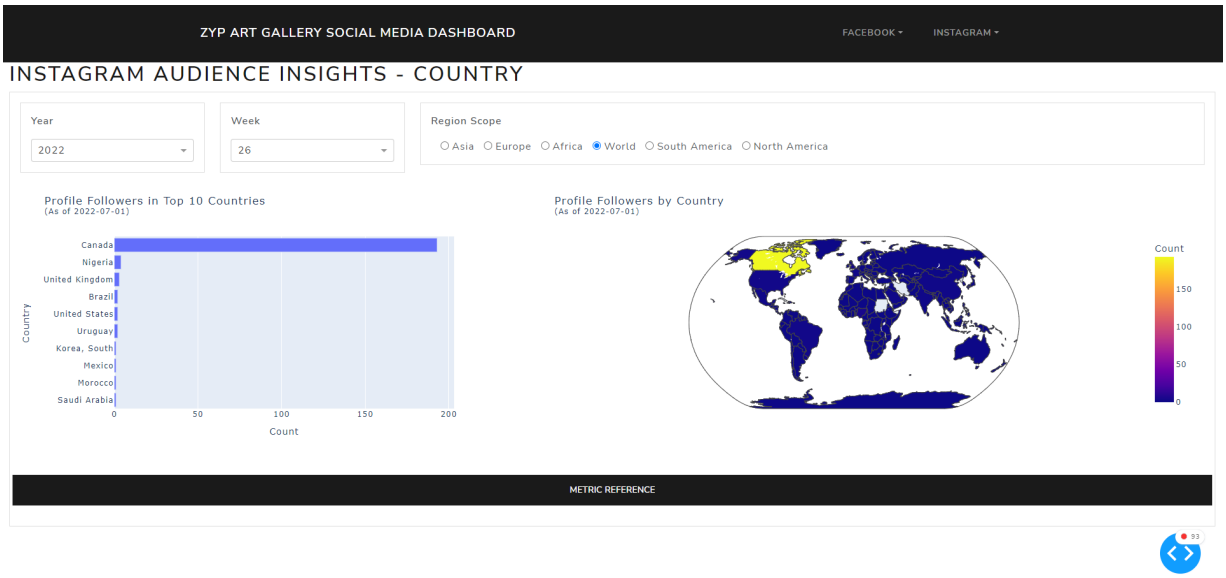
Data

This webpage uses the “ZypInstagram_Audience-Age&Gender” Google Sheets document that is saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
end_time	Date
year	The year the data was extracted (retrieved from end_time)
week	The week in the year the data was extracted (retrieved

	from end_time)
Metric	***Title & Description***
page_fans_gender_age [Gender abbreviation].[Age interval] E.g. F.13-17 (Female aged 13 to 17) • F = Female • M = Male • U = Unknown	*Gender and Age* The gender and age distribution of this profile's followers

Instagram Audience Insights - Country



The above screenshot shows the Instagram Audience Insights - Country webpage. The intent of this page is to communicate the performance of the organization's Instagram page by country.

<u>Interactive elements & Visualization</u>

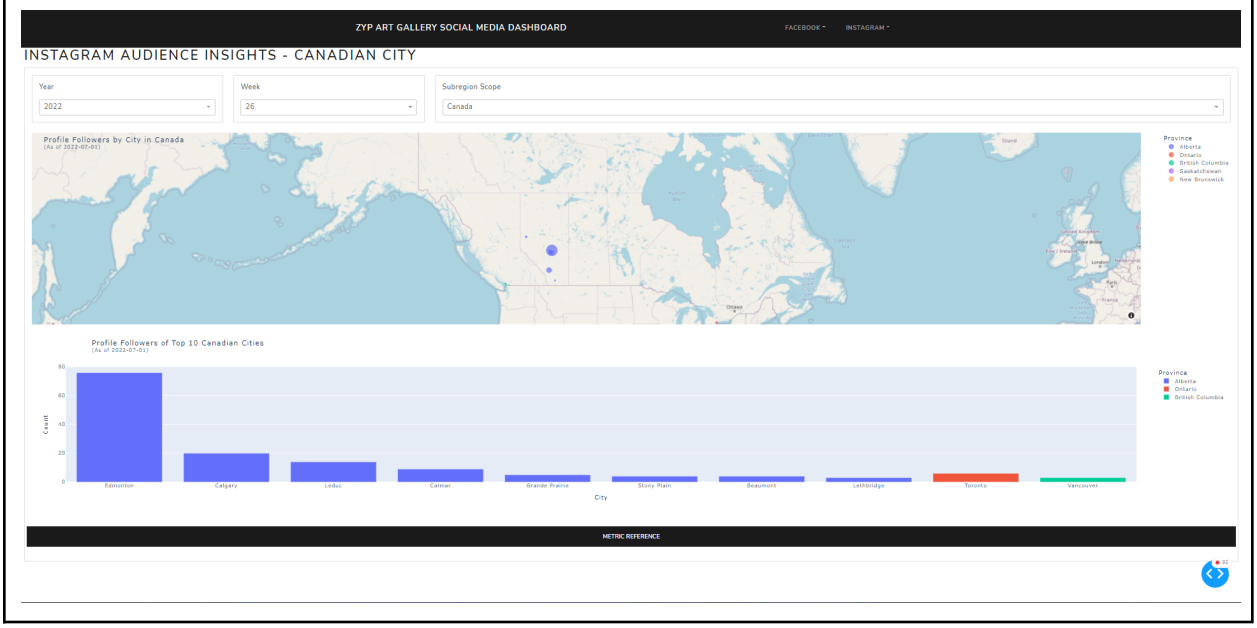
As touched upon earlier in this section of the report, all audience related metrics are of period Lifetime per week. Thus, the daterange filter is replaced by the year & week dropdown filter. It functions in the same way as the daterange filter. The region scope filter function in a similar way as that of the corresponding Facebook section. Changing the selected option in the year and/or week dropdown filters would change the choropleth and top 10 country bar chart accordingly to reflect the metric value as of the selected year & week. The region scope filter would also further adjust the visualizations accordingly. By default the current year and prior to the current week is selected.

Data

This webpage uses the “ZypInstagram_Audience-Country” Google Sheets document that is saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
end_time	Date
year	The year the data was extracted (retrieved from end_time)
week	The week in the year the data was extracted (retrieved from end_time)
Metric	***Title & Description***
audience_country Country name => E.g. South Africa	*Audience Country* The countries of this profile's followers

Instagram Audience Insights - Canadian City



The above screenshot shows the Instagram Audience Insights - Canadian City webpage. The intent of this page is to communicate the performance of the organization’s Instagram page by Canadian city.

Interactive elements & Visualization

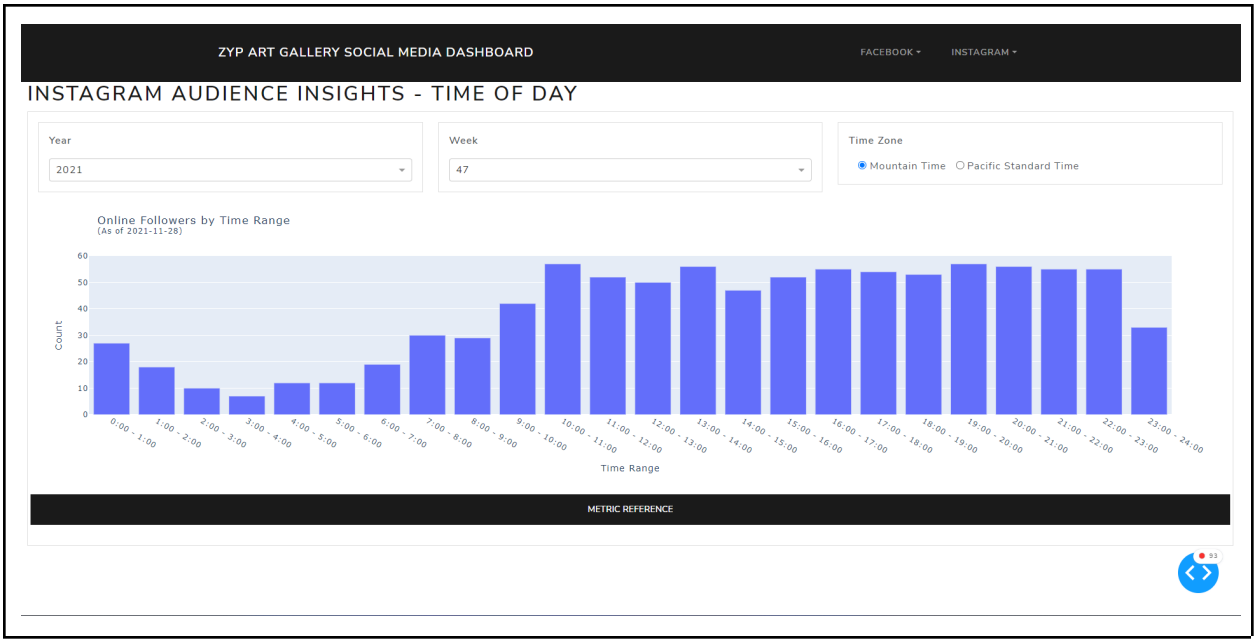
As touched upon earlier in this section of the report, all audience related metrics are of period Lifetime per week. Thus, the daterange filter is replaced by the year & week dropdown filter. It functions in the same way as the daterange filter. The subregion scope filter function in a similar way as that of the corresponding Facebook section. Changing the selected option in the year and/or week dropdown filters would change the bubble map chart and top 10 Canadian city bar chart accordingly to reflect the metric value as of the selected year & week. The subregion filter would also further adjust the visualizations accordingly. By default the current year and prior to the current week is selected.

Data

This webpage uses the “ZypInstagram_Audience-CanadianCity” Google Sheets document that is saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
end_time	Date
year	The year the data was extracted (retrieved from end_time)
week	The week in the year the data was extracted (retrieved from end_time)
Metric	***Title & Description***
audience_city City name, Province => E.g. Calgary, Alberta	*Audience City* The cities of this profile's followers

Instagram Audience Insights - Time of Day



The above screenshot shows the Instagram Audience Insights - Time of Day webpage. The intent of this page is to communicate the performance of the organization's Instagram page by time of day.

Interactive elements & Visualization

As touched upon earlier in this section of the report, all audience related metrics are of period Lifetime per week. Thus, the daterange filter is replaced by the year & week dropdown filter. It functions in the same way as the daterange filter. The time zone scope filter function in a similar way as that of the corresponding Facebook section. Changing the selected option in the year and/or week dropdown filters would change the bar chart accordingly to reflect the metric value as of the selected year & week. The time zone filter would also further adjust the visualizations accordingly. By default the current year and prior to the current week is selected.

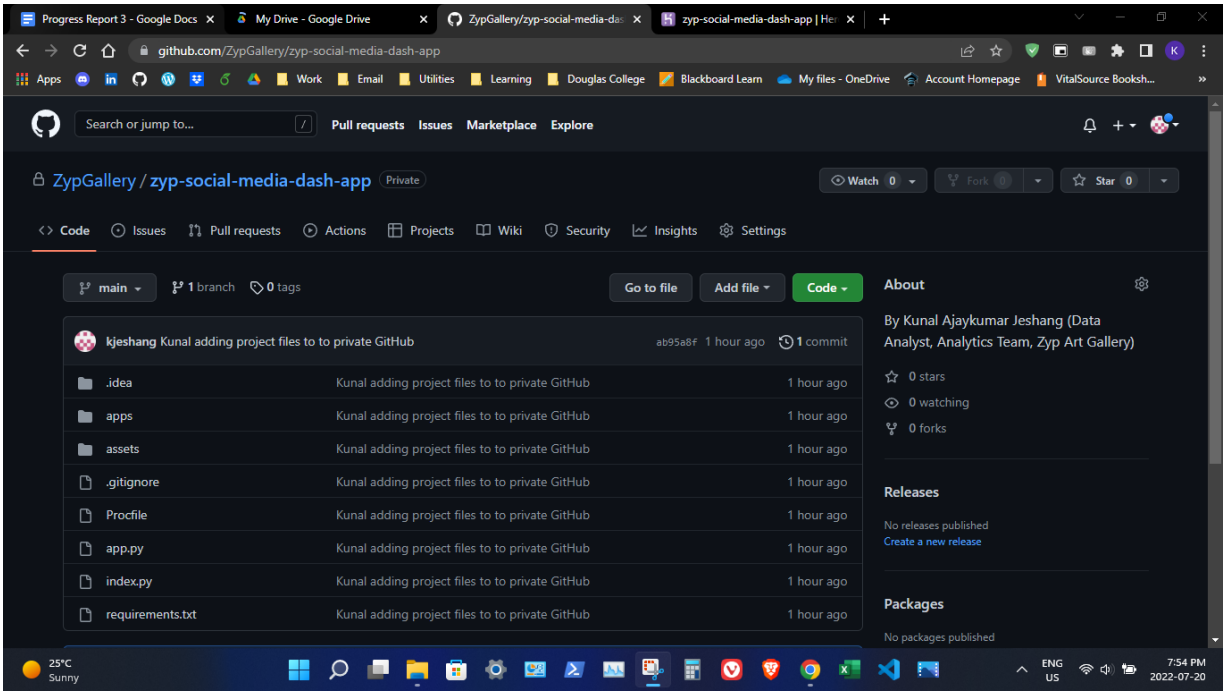
Data

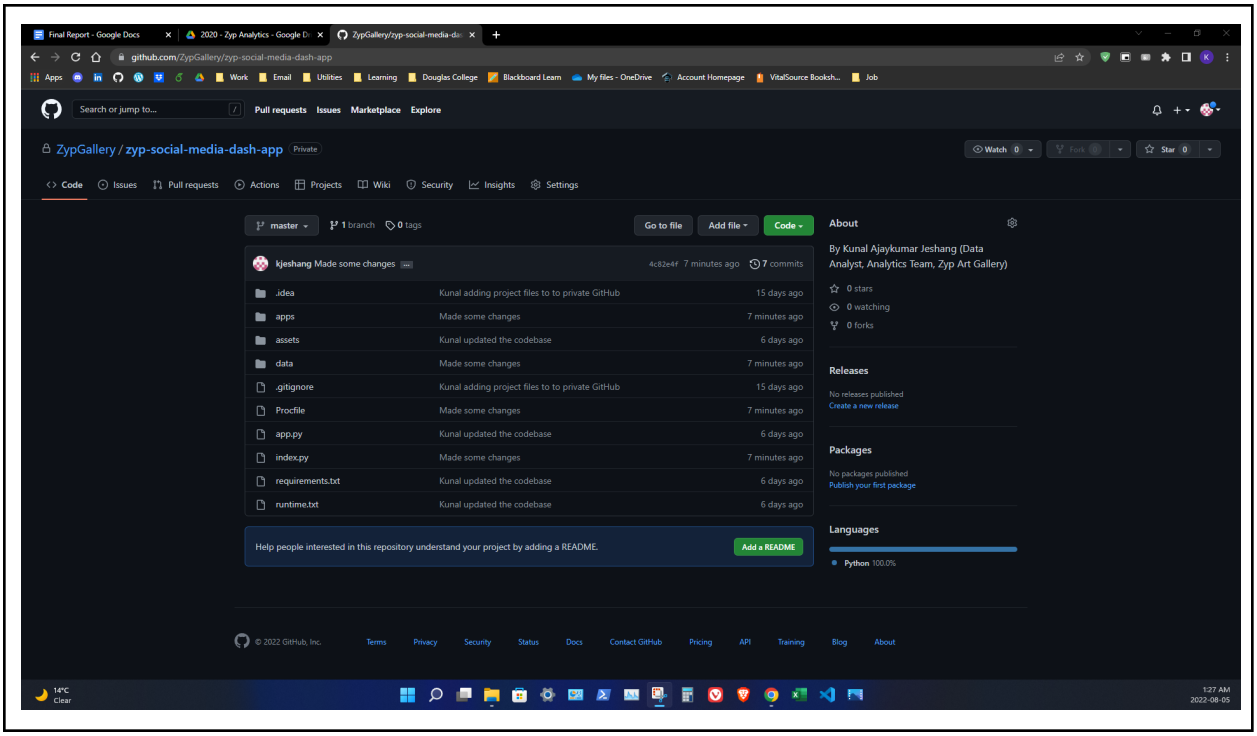
This webpage uses the "ZypInstagram_Audience-TimeOfDay" Google Sheets document that is saved on the Organization's Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
end_time	Date
year	The year the data was extracted (extracted from end_time)
week	The week in the year the data was extracted (extracted from end_time)

Metric	***Title & Description***
online_followers Time range in the 24 hour format => E.g. 14:00 - 15:00	*Audience City* The cities of this profile's followers

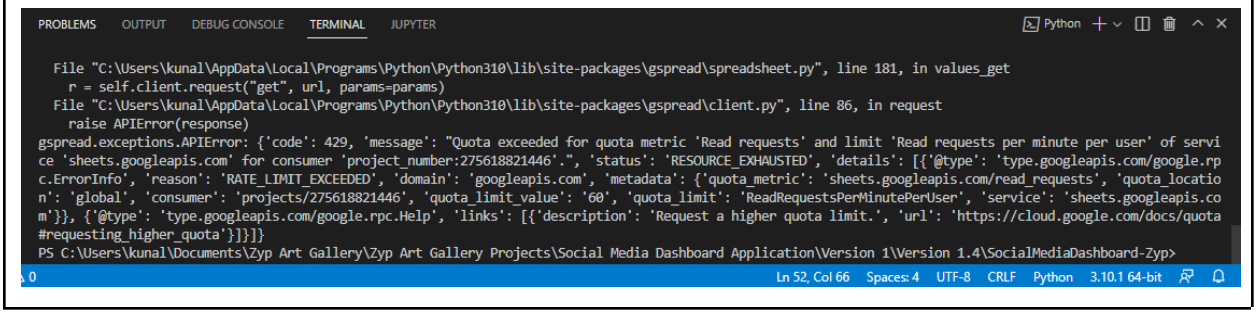
Challenges & Updates

Proof of project uploaded to organization GitHub (July 20th 2022 7:54PM)

Proof of project uploaded to organization GitHub (August 5th 2022 1:27PM)



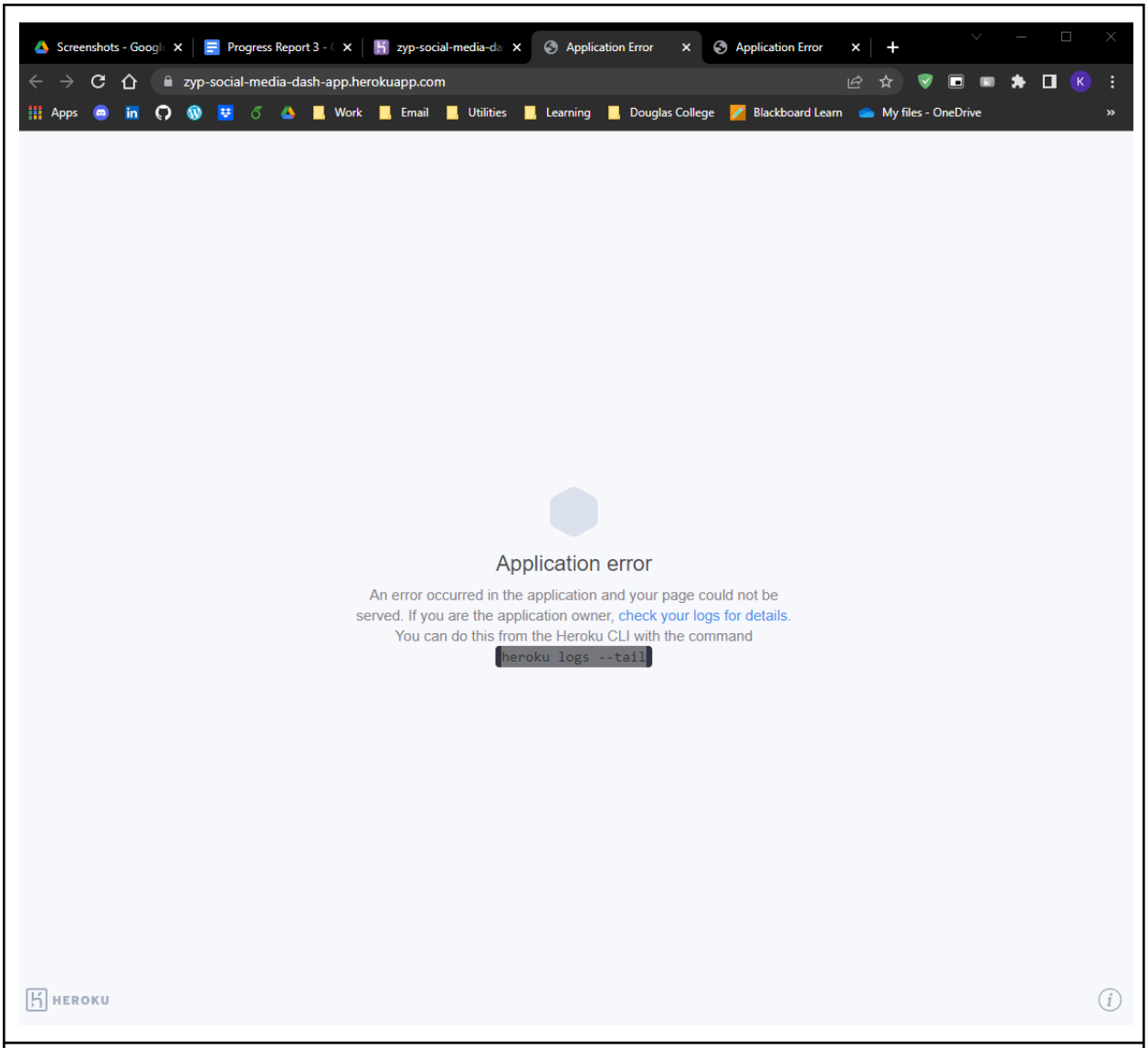
The above screenshot shows proof that I uploaded the project to the organization's private GitHub. This way, the Analytics team members can pull the project from the repository and run it on their own computers.

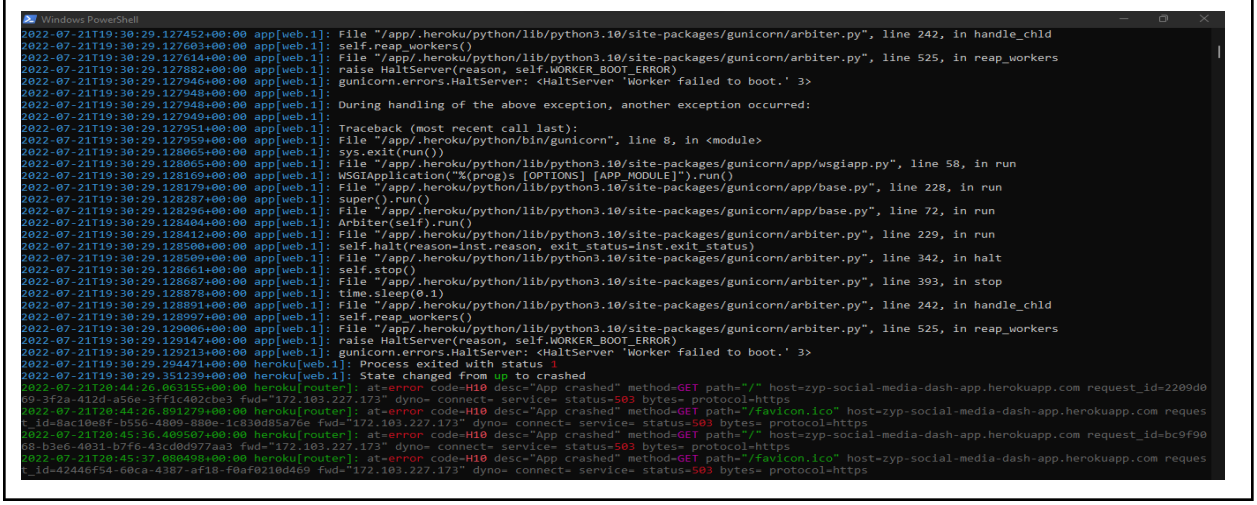
Proof of Google Sheets API threshold error



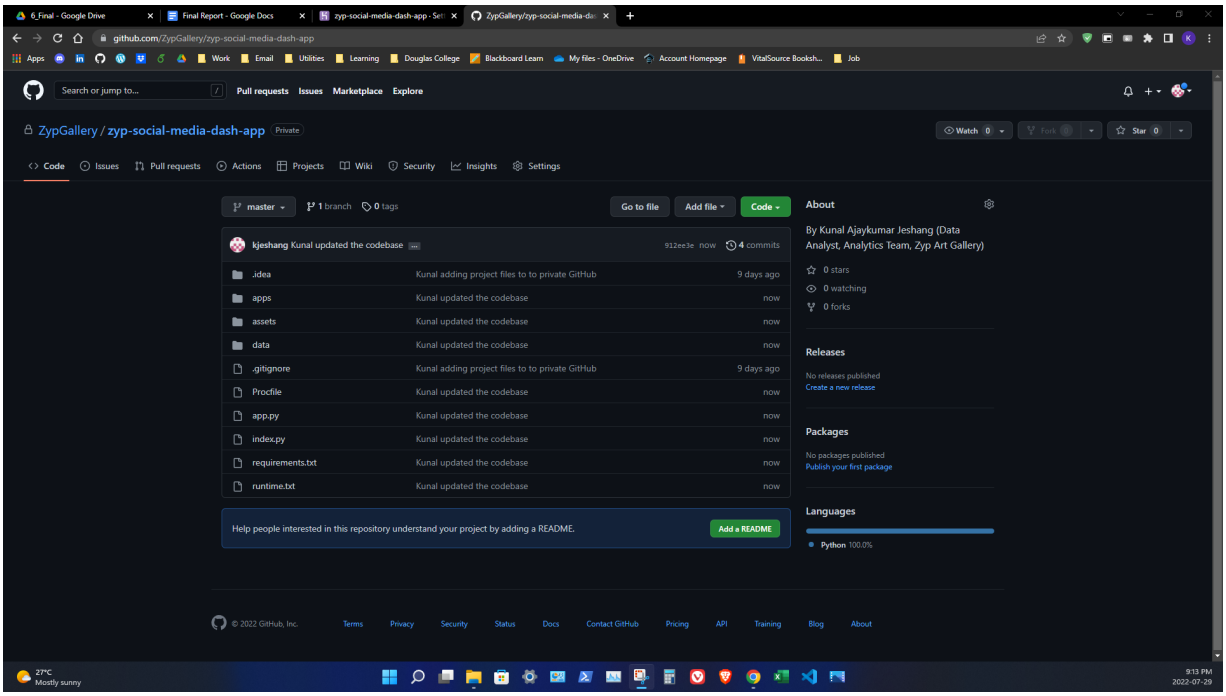
The above shows proof that of the Google Sheets API request threshold limit error which stops the application from running if at has been started-and-stopped too many times, in turn, importing data from Google Drive consequently too much. This error message was displayed when attempting to run the application on a local server.

Proof of Heroku Application error


Screenshots - Google
Progress Report 3 -
zyp-social-media-da
Application Error
Application Error
zyp-social-media-dash-app.herokuapp.com
Apps
Work
Email
Utilities
Learning
Douglas College
Blackboard Learn
My files - OneDrive
Application error
An error occurred in the application and your page could not be served. If you are the application owner, check your logs for details.
You can do this from the Heroku CLI with the command
heroku logs --tail
HEROKU


Windows PowerShell
2022-07-21T19:30:29.127452+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 242, in handle_chld
2022-07-21T19:30:29.127603+00:00 app[web.1]: self.reap_workers()
2022-07-21T19:30:29.127614+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 525, in reap_workers
2022-07-21T19:30:29.127882+00:00 app[web.1]: raise HaltServer(reason, self.WORKER_BOOT_ERROR)
2022-07-21T19:30:29.127946+00:00 app[web.1]: gunicorn.errors.HaltServer: <HaltServer 'Worker failed to boot.' 3>
2022-07-21T19:30:29.127948+00:00 app[web.1]:
2022-07-21T19:30:29.127948+00:00 app[web.1]: During handling of the above exception, another exception occurred:
2022-07-21T19:30:29.127949+00:00 app[web.1]:
2022-07-21T19:30:29.127951+00:00 app[web.1]: Traceback (most recent call last):
2022-07-21T19:30:29.127959+00:00 app[web.1]: File "/app/.heroku/python/bin/gunicorn", line 8, in <module>
2022-07-21T19:30:29.128065+00:00 app[web.1]: sys.exit(run())
2022-07-21T19:30:29.128065+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/app/wsgiapp.py", line 58, in run
2022-07-21T19:30:29.128169+00:00 app[web.1]: WSGIApplication("%(prog)s [OPTIONS] [APP_MODULE]").run()
2022-07-21T19:30:29.128179+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/app/base.py", line 228, in run
2022-07-21T19:30:29.128287+00:00 app[web.1]: super().run()
2022-07-21T19:30:29.128296+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/app/base.py", line 72, in run
2022-07-21T19:30:29.128404+00:00 app[web.1]: Arbiter(self).run()
2022-07-21T19:30:29.128412+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 229, in run
2022-07-21T19:30:29.128500+00:00 app[web.1]: self.halt(reason=inst.reason, exit_status=inst.exit_status)
2022-07-21T19:30:29.128509+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 342, in halt
2022-07-21T19:30:29.128661+00:00 app[web.1]: self.stop()
2022-07-21T19:30:29.128687+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 393, in stop
2022-07-21T19:30:29.128878+00:00 app[web.1]: time.sleep(0.1)
2022-07-21T19:30:29.128891+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 242, in handle_chld
2022-07-21T19:30:29.128997+00:00 app[web.1]: self.reap_workers()
2022-07-21T19:30:29.129006+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 525, in reap_workers
2022-07-21T19:30:29.129147+00:00 app[web.1]: raise HaltServer(reason, self.WORKER_BOOT_ERROR)
2022-07-21T19:30:29.129213+00:00 app[web.1]: gunicorn.errors.HaltServer: <HaltServer 'Worker failed to boot.' 3>
2022-07-21T19:30:29.294471+00:00 heroku[web.1]: Process exited with status 1
2022-07-21T19:30:29.351239+00:00 heroku[web.1]: State changed from up to crashed
2022-07-21T20:44:26.063155+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/" host=zyp-social-media-dash-app.herokuapp.com request_id=2209d0
69-3f2a-412d-a56e-3ff1c402cbe3 fwd="172.103.227.173" dyno= connect= service= status=503 bytes= protocol=https
2022-07-21T20:44:26.891279+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/favicon.ico" host=zyp-social-media-dash-app.herokuapp.com reques
t_id=8ac10e8f-b556-4809-880e-1c830d85a76e fwd="172.103.227.173" dyno= connect= service= status=503 bytes= protocol=https
2022-07-21T20:45:36.409507+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/" host=zyp-social-media-dash-app.herokuapp.com request_id=bc9f90
68-b3e6-4031-b7f6-43cd0d977aa3 fwd="172.103.227.173" dyno= connect= service= status=503 bytes= protocol=https
2022-07-21T20:45:37.080498+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/favicon.ico" host=zyp-social-media-dash-app.herokuapp.com reques
t_id=42446f54-60ca-4387-af18-f0af0210d469 fwd="172.103.227.173" dyno= connect= service= status=503 bytes= protocol=https

The screenshots above show that despite successful deployment, the application still was not rendered on Heroku. As per the instructions on the first screenshot, I used the “heroku logs --tail - -app <app name>” command to find out what errors occurred. The errors can be seen in the second screenshot.

Proof of project uploaded to organization GitHub (July 29th 2022 9:13PM)


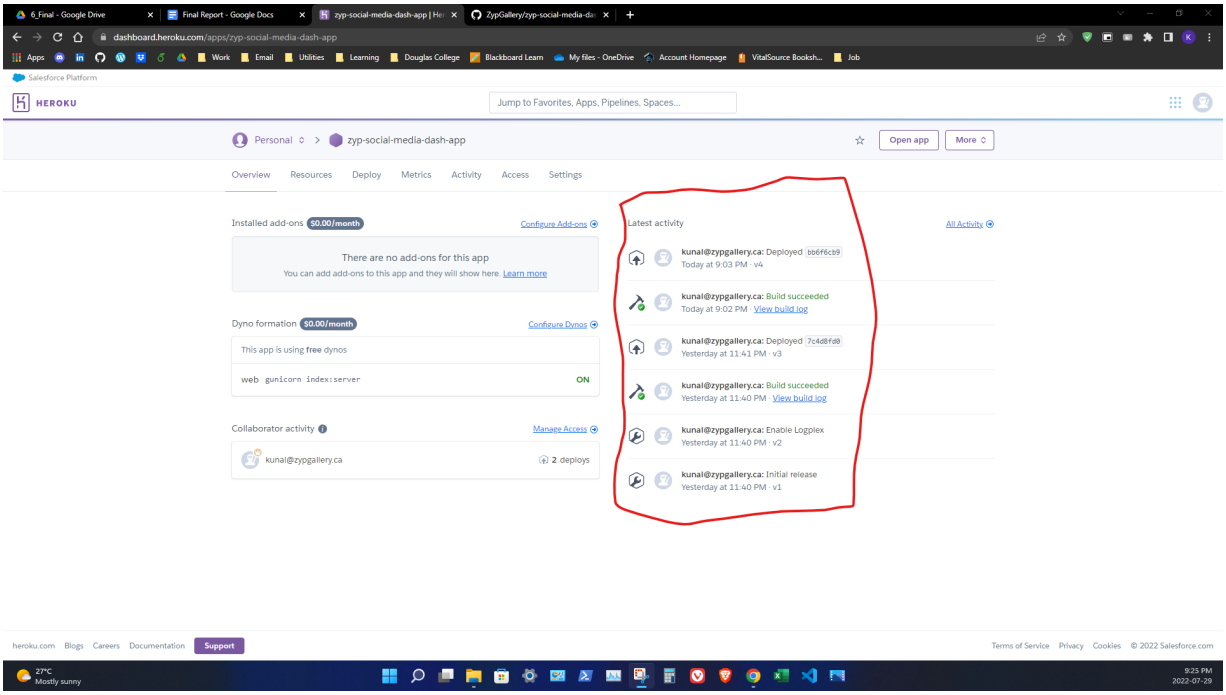
The above screenshot shows proof that I pushed the latest working version of the project on the organization GitHub. The version that was pushed at the above mentioned time reflects the changes made to the application such that it was successfully deployable to Heroku. Below are the changes I had to make to deploy the project successfully to Heroku without any post-deplyment application errors.

- The main change that enabled the Dash application to deploy was reverting to using simple CSV files saved on the project repository rather than directing importing the contents of social media data Google Sheets document into the application dynamically using Google Sheets API & the gspread package
- The Flask package’s functions were used to instantiate the Dash application and server in the “app” Python script rather than the respective Dash functions
    - I referred to the following Stack Overflow link to make the change. Although, I am not sure why this change helped the application to deploy: https://stackoverflow.com/questions/61452429/plotly-dash-app-gives-error-after-deployment-to-heroku
- According to the Heroku logs, there was some form of favicon error. Specifically, the Heroku log infers that a favicon icon file is required by a Dash application that is deployed to Heroku. Thus, I took the Zyp Art Gallery’s logo (downloaded from the

organization's Google Drive) and converted it from PNG file format to ICO file format. Then, I changed the name of the newly converted logo to "favicon", and placed in the "assets" folder of the project directory.

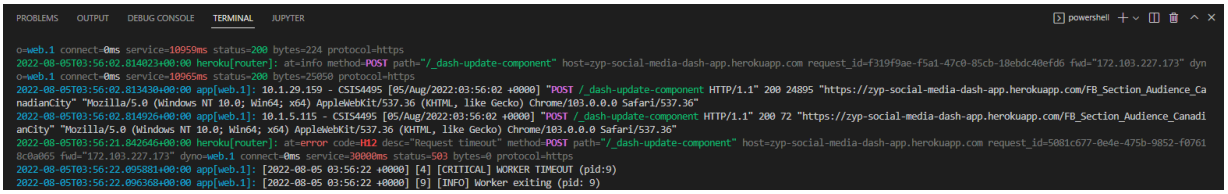
- I used the following link to convert the Zyp Art Gallery logo from PNG to ICO: https://favicon.io/favicon-converter/
- The links below helped me understand the Heroku log referring to the "Favicon.ico".
  - https://community.plotly.com/t/display-favicon-in-heroku-dash-app/44013
  - https://github.com/dc-aichara/DS-ML-Public/tree/master/Medium_Files/dashboard_demo

Proof of successful first-time deployment and deployment of changes

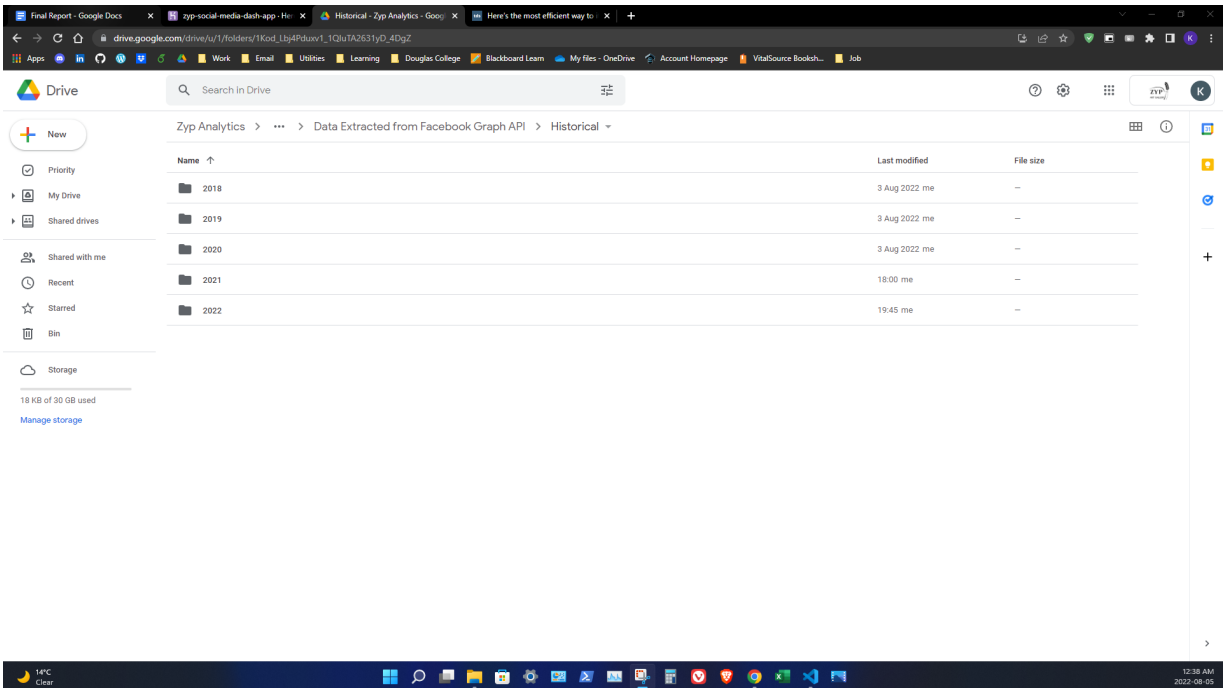


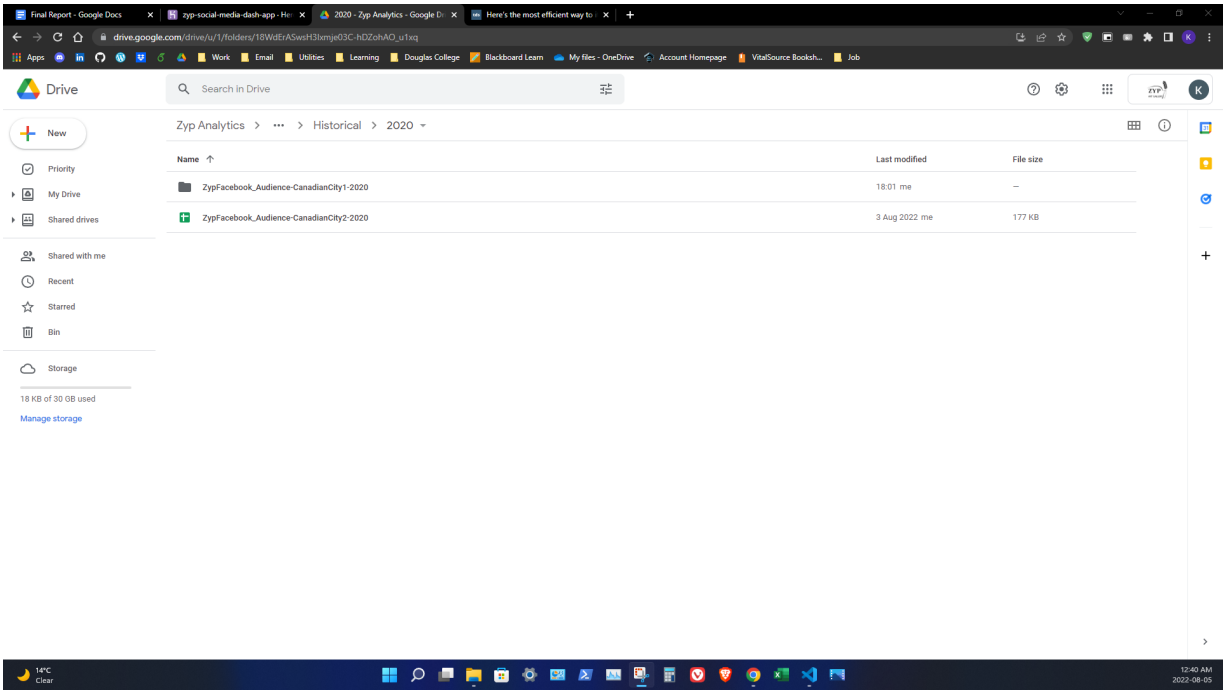
The above screenshot shows proof that I successfully managed to deploy the project to Heroku as a brand new application (i.e., see "kunal@zypgallery.ca: Deployed 7c4d8fd0). The screenshot also indicates successful deployment of the application with changes made to the project, which is mainly using updated CSV files (i.e., kunal@zypgallery.ca: Deployed bb6f6cb9).

Request timeout error caused by very large "ZypFacebook_Audience-CanadianCity1" CSV file

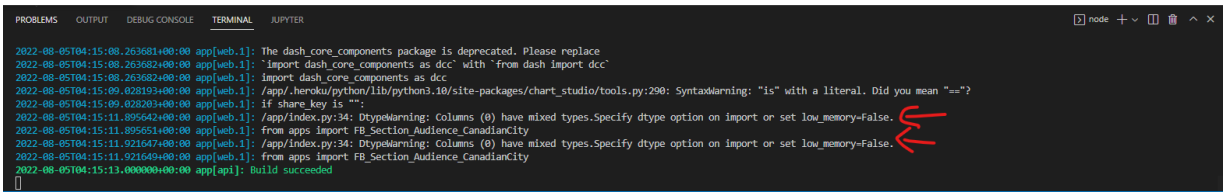


Backing up past years Facebook Canadian City data to Google Drive

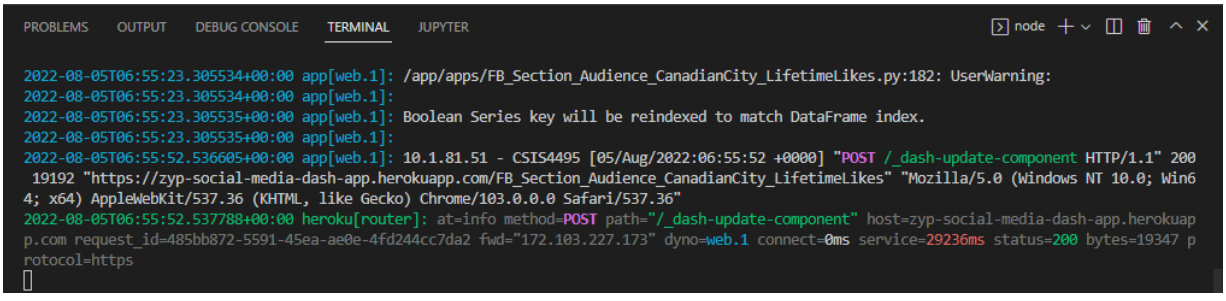




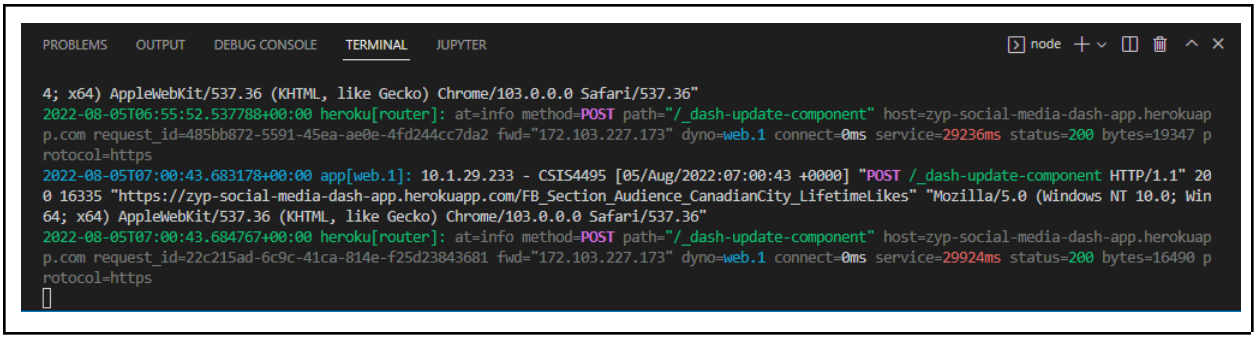
Strange message regarding the Facebook Canadian City script in app folder



This is one of the few instances when the request did not timeout to render the visualizations related to “ZypFacebook_Audience-CanadianCity1” dataset with subregion scope of ‘Canada’.



This is one of the few instances when the request did not timeout to render the visualizations related to “ZypFacebook_Audience-CanadianCity1” dataset with subregion scope of ‘Alberta’.

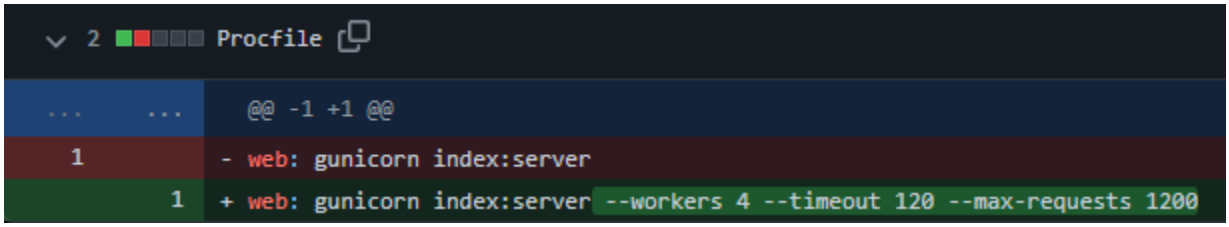


I attempted to follow the advice stated in the following links but I was unsuccessful in addressing the issue. There were few instances where the visualizations related to the "ZypFacebook_Audience-CanadianCity1" dataset (i.e., Facebook Canadian City Lifetime Likes) managed to render. This was when the request completed within just less than 30 seconds (i.e., 30,000 milliseconds). Unfortunately, more often than not, the visualizations did not render.

- https://dash.plotly.com/dash-enterprise/troubleshooting
- https://stackoverflow.com/questions/10855197/gunicorn-worker-timeout-error
- https://community.plotly.com/t/dash-heroku-timeout/36517/4
- https://stackoverflow.com/questions/60537977/critical-worker-timeout-in-logs-when-running-hello-cloud-run-with-python-f
- https://devcenter.heroku.com/articles/preventing-h12-errors-request-timeouts
- https://devcenter.heroku.com/articles/python-gunicorn#worker-timeouts

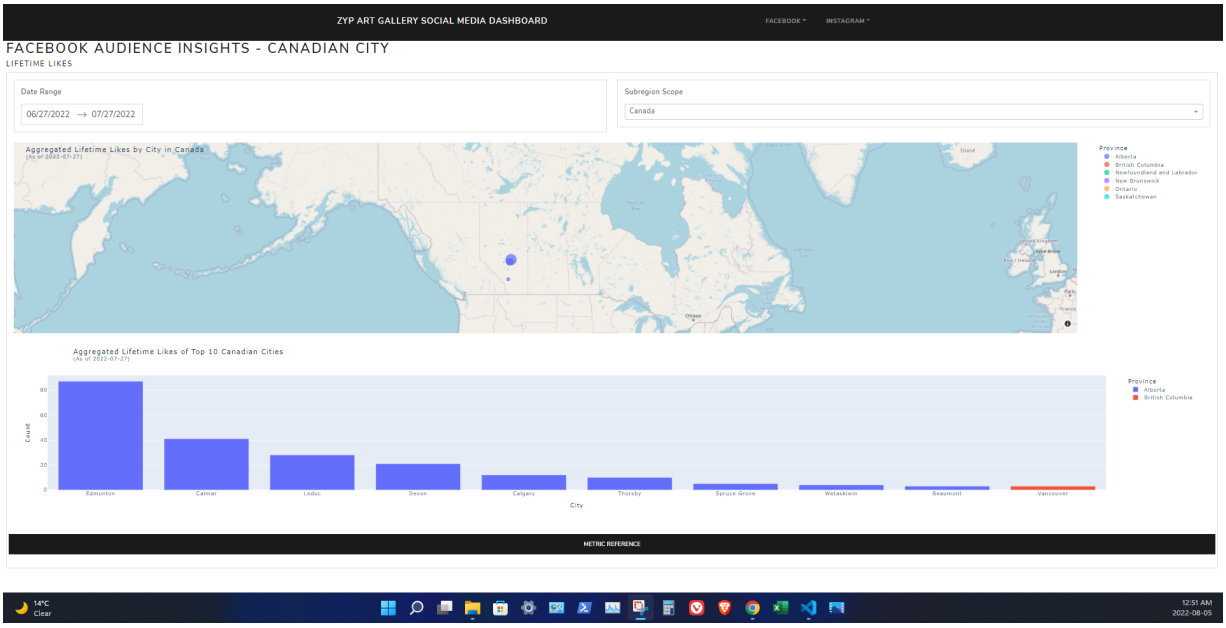
I tried to address this issue in the following ways:

- Made changes to the Procfile by adjusting guinicorn settings.

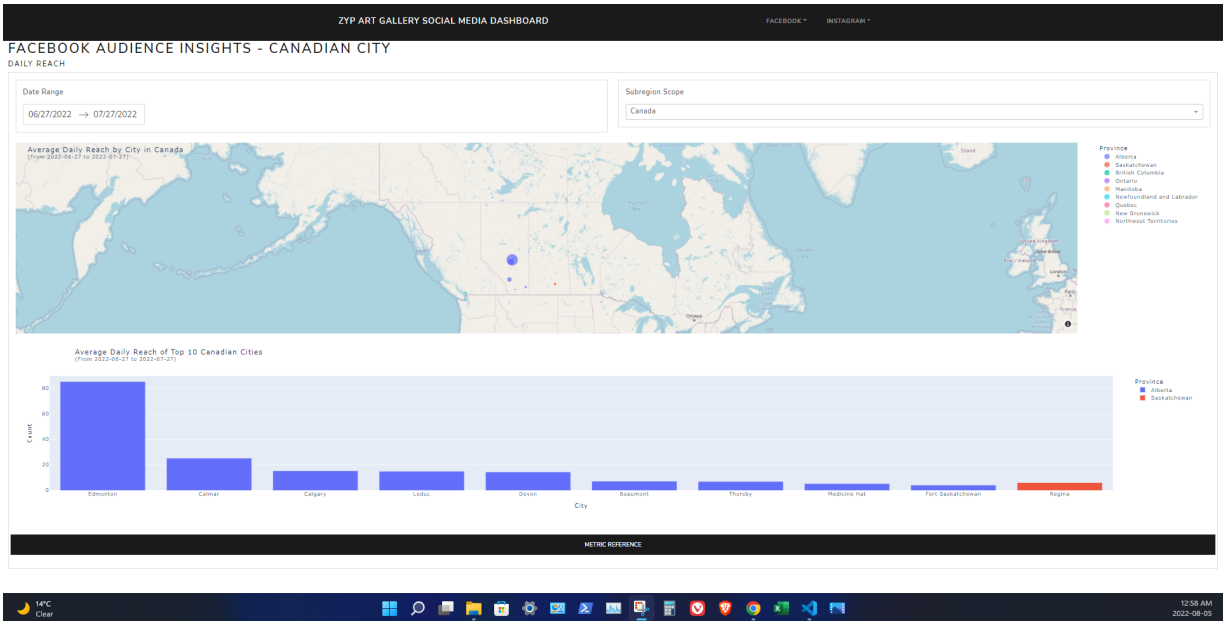


- Reducing the sizes of "ZypFacebook_Audience-CanadianCity1" and "ZypFacebook_Audience-CanadianCity2" datasets to only include data from the current quarter of 2022. I backed up the past years data to the organization's Google Drive. Although, I was still experiencing inconsistent results.
- Splitting the Facebook Canadian City section into two separate sections. One section dedicated to Facebook Canadian City Lifetime Likes (which uses "ZypFacebook_Audience-CanadianCity1" dataset), and another section dedicated to Facebook Canadian City Daily Reach (which uses ZypFacebook_Audience-CanadianCity2 dataset). I had limited success. The section regarding Facebook Canadian City Daily Reach rendered very quickly, but the Facebook Canadian City Lifetime Likes still had inconsistent results. Although, the chance of rendering the visualizations for Facebook Canadian City Lifetime Likes was much higher due to the aforementioned change compared to having a unified Facebook Canadian City section.

Facebook Audience Insights - Canadian City (Lifetime Likes)
when it does render within 30 seconds



Facebook Audience Insights - Canadian City (Daily Reach)



These new sections function the same way as the older one, but the sets of visualizations are not joined together in a singular page via a tab. They are separated and can be accessed via the navigation bar.

Scripts

The link below is the GitHub repository of the Zyp Art Gallery social media extraction code. It is on my public GitHub profile. There is fully functioning version of my extraction code on the organization's GitHub profile but it is private. The version on my GitHub profile does not contain the actual access token or Google service account private key. This is to protect the data privacy of the organization. That being said, a working version of the social media data extraction code would be provided in the Blackboard submission.

Zyp Art Gallery Social Media Data Extraction Codebase:
https://github.com/kjeshang/ZypArtGallery-SocialMediaDataExtraction

Below is the GitHub repository of the custom dashboard application itself from my personal GitHub profile. The actual Google service account private key is not provided to protect the organization's data privacy. The social media data files are also not present here. Thus, the project on my personal GitHub profile not a version that can be run locally; it is intended for viewing the codebase only. That being said, the last deployed version of the project that was deployed to Heroku would be provided in the Blackboard submission.

Zyp Art Gallery Social Media Dashboard Codebase:
https://github.com/kjeshang/ZypArtGallerySocialMediaDashboard

Deployed Application

Below is the link of the actual application deployed online via Heroku. Login credentials provided to instructor in Blackboard submission.

Zyp Art Gallery Social Media Dashboard Heroku Link:
https://zyp-social-media-dash-app.herokuapp.com/

Video Presentation

Below is the link to the video presentation. The video is saved on my Douglas College OneDrive. However, it can be accessed by anyone who has the link and is part of the Douglas College organization. Few portions of the presentation were receycled from the Midterm Video Presentation albeit shorentened or removed entirely. Most portions are entirely new, or re-recorded to reflect further clarity and changes. The presentation was recorded using the free Panopto Screen Recorder, and was edited using the Windows built-in Video Editor. Video recording and editing is not my strongest skill, but I tried my best. Also, during the time of recording I had a sore throat. Thus, I apologize for my breathlessness and low voice during some parts of the presentation. That being said, I tried my best to speak clearly & concislely. Also, I was not able to cover all filteration features in the sections of the application due to the wide scope of the application itself, and the 15 minute time limit. In certain portions of the project demonstration, I chose to edit out parts where I am simply toggling between one section to

another using the navigation bar. This is particularly the case with the Facebook Sections as they typically take longer to load. Thus, I tried my best to cover the main features and show all sections as concisely as possible.

Final Video Presentation Link:
https://collegedouglas-my.sharepoint.com/:v:/g/personal/jeshangk_student_douglascollege_ca/EeKl-_CMWQRCl9fzYt6Ra80Bje244dNnFDYsAYFVQln68g?e=B9Hx1t

Addressing Project Objectives

Is the application easy to access by those that are part of the Zyp Art Gallery organization?

Despite performing the appropriate research and following the correct steps based on the research, I was unable to make the application accessible to everyone that is part of the organization in the way I originally intended. Specifically, I was facing challenges deploying the application to Heroku. I tried my best to follow the instructions from the sources I researched but even after successful deployment of the application, I faced application error/s when opening the Heroku link to the project. More explanation regarding the challenges and errors are discussed in the Progress Report 3 part in the Appendix.

That being said, I eventually did manage to deploy to Heroku but only after making some significant changes. I initially intended for the project to directly import the social media data via Google Sheets API and the gspread package. However, due to the great size of the social media data, post-deployment of the project lead to application errors, which was primarily caused by Google Sheets API request threshold being reached. When I reverted to using social media data in the form of CSV files that are saved in the project directory, the application managed to deploy successfully. Therefore, the design logic of my application was altered to make the application deployable. Further changes I needed to make are discussed in the previous section of the report. Also, to be able to access the application, one would require the appropriate username and password as the application utilizes basic HTTP authentication for security purposes. The login credentials would be provided by me or any other member of the Zyp Art Gallery Analytics team.

Is the application easy to use even for a non-technical person?

As my target group is primarily the Zyp Art Gallery Analytics team, to a certain degree they are all technical people but of all varied levels. Although, even if a non-technical person were to use the application, they would not face significant learning curves. This is because the project is a web application and has near-identical functionality to that of a website.The Analytics team leader did feel that the User Interface overall could be improved although he is satisfied in its current form because as of now, the project was developed individually. Post-submission of the project for the CSIS 4495, there would prospectively be further discussion and contribution from the rest of the team on how to improve the User Interface and User Experience.

The sections that visualize Facebook Canadian City data tends to load slow or unfortunately not render successfully. When running the application on a local server, these sections would take anywhere between 1 to 2 minutes. When running the application on Heroku (on the free tier), these sections would either manage to render within 30 seconds, or the request would timeout and the section would not render at all. This is due to the fact that the datasets used in these sections are very large. Specifically, these datasets have 9076 columns, where there is one column refers to the date and the remaining 9075 columns refers to a Canadian city, town, municipality, etc. This negatively affects the user experience for my primary target audience. This made me realize the pros and cons of dedicated dashboard softwares versus my project. Plotly Dash and Heroku are great free tools that allowed me to create a custom dashboard application that does not require my target audience to install special software and learn new skills. Although, the technology does not have the power to process datasets of a large size optimally.

Is the application overall aesthetically pleasing?

The application is somewhat pleasing to the eye. There has not been any comment by the Analytics team during the meetings regarding the aesthetics of the application itself. Thus, I can presume that the application's theme and color palette are sufficient for now. The LUX theme is used by the Plotly Dash application which has mostly black and white color palette. This matches the theme and logo of Zyp Art Gallery, which uses white, black, and shades of grey.

Does the application follow a logical design in regards to navigation?

The application does follow a logical design. The project files are split across an "index" Python script that initializes the entire application (on a local machine) and renders the home page. There is an "app" Python script that instantiates the application itself, and the variable is called within the index script. There is an assets folder that has files that act to import datasets and hold reference data. There is an app folder that contains Python scripts that render the various webpages of the application. The "index" script creates the skeleton of the various sections of the dashboard application. A consistent aspect of all of the various sections of the application is the navigation bar. If one needs to transition to another section of the application, it is by selecting one of the links from one of the dropdown menus on the navigation bar. There is a dropdown menu on the navigation bar for both Facebook and Instagram. In terms of the code, the "index" script retrieves the content of a section of the application based on the corresponding link selected from the navigation bar. Also, the title of the application also appears on the navigation bar, and it is linked to the home page. In a sense, everything leads to the "app" and "index" scripts that are part of the project's root directory. Therefore, the logical design of the application is logical in terms of both project structure and user interactivity.

Are the visualizations detailed & informative, yet easy on the eyes?

The visualizations created at this current time are detailed and informative enough according to the Zyp Art Gallery Analytics team members. I took inspirations from my time learning about

social media data used by Facebook & Instagram, and constructing extraction code. Specifically, I was researched the ‘Later’ platform and ‘Facebook for Business’ to find out what metrics these platforms are using, in turn, I was exposed to the visualizations as well. As aforementioned, prior to the start of the CSIS 4495 course, I was mostly in charge of social media data extraction and other senior members of the team constructed visualizations using Power BI/Tableau. Thus, the dashboards my fellow team members made also served as inspiration. All of the aforementioned helped me to make decisions on the choice of visualizations in this application, but at the same time addressed the lack of filtration and granularity for the purpose of drawing out the maximum-possible informativity. Thus, I have met the required expectations of my primary audience.

Do the visualizations have a good degree of interactivity in terms of date range, category, and/or any other identifier?

Yes it does. This is proven by providing maximum filtration to adjust the date range via datepickers, metric category selections handled by checklists, radio buttons, and dropdowns. In turn, the visualizations respond to the changes made to the aforementioned interactive elements and adjust accordingly. Certain visualizations utilizing a large dataset, such as those related to Country and Canadian City, may render slower which may be inconvenient. It should be noted that these visualizations do render relatively faster when the application is accessed via Heroku rather than running it on a local server. The Zyp Art Galery Analytics team is understanding regarding this and remain statisfied regarding the overall level of interactivity of all the visualizations in the application.

Environment Setup

Environment Setup to extract Social Media data

As aforementioned in earlier sections of this report, I have been volunteering at Zyp Art Gallery as of June 2021. Most of my work involved extracting Facebook and Instagram data. The basis of this project involves analyzing and visualizing the extracted social media data in a easy-to-use and accessible way. Thus, without having access to the social media data, the scope of this project may have been to wide to complete within a semester. The social media data itself was extracted using Python programming with the help of Facebook Graph API. In order to perform requests to Facebook Graph API, I needed to have a Facebook account and have a page role in the organization’s Facebook page. I ended up making a Facebook account using my organization email. After creating a Facebook account, the executive director then assigned my Facebook profile to have a page role on the organization’s web page. I was then able to create a ‘Facebook for Developers’ account. I then had to create an ‘app’ on Facebook for Developers, and then state the purpose of it which was simply to make requests. To gain full access to information provided by various metrics, I had to provide Zyp Art Gallery’s privacy policy for approval before gaining full access.

Even though Facebook for Developers allows for making requests within the Graph Explorer, the drawback was that some request output was too long thus pagination would occur. In other words, the entire output would not be accessible within the explorer without further manipulation. This is further exemplified by the fact that the data output is in JSON format. Thus, making the API request with the help of Python programming was easier because I could code a loop that would paginate through the JSON output. In order to make requests to Graph API via a Python script, I needed to have an access token. Although, an access token is required overall to make requests to Graph API. Within Graph Explorer, an access token is provided by default but it is only valid for approximately two hours. Thus, using a short-lived access token is not ideal when running social media data extraction code on a weekly basis. And so, I extended the access token to last approximately two months. Unfortunately, Facebook for Developers do not provide lifelong access tokens. Thus, after the current token expires, I generate a new token and extend its lifetime. An access token with a longer lifetime was more ideal for me to create the social media data extraction code. This makes the act of running the social media data extraction code on a weekly basis easily executable on a weekly basis as I not have to keep on generating a new access token via the Graph Explorer. I only need to generate an access token with an extended lifetime every so often.

The organization utlilzes Google Drive as the primary office suite and data storage space at this current time. During the early days after I completed creating social media data extraction scripts for posts, page, & demographic insights, my final output would be a collection of data files in ‘CSV’ format. I used to manually upload them to a folder on the organization’s Google Drive. This used to be quite a tedious task. Thus, I decided to create a Google Service Account via the Google Cloud console. I then added the Google Drive and Google Sheets API. I then was able to acquire a private service account key which allows me to interact with Google Drive & Google Sheets. The next step was to create a dedicated Google Sheets file with a singular sheet that corresponds with the name of the data files in ‘CSV’ file format. The name of the Google Sheets file, sheet name, and identification value of the Google Sheets file (which is found in the URL), are utilized with the Google service account private key to take the contents of the data files in ‘CSV’ file format, and paste it into the respective Google Sheets files. The Python package called ‘gspread’ is imperative to auto-inserting the data to Google Sheets files.

As aforementioned, the social media data extraction code was created using Python. It was coded using VS Code. There are multiple scripts that are dedicated to extracting different types of metrics, yet they all follow a similar process with slight variations in terms of data cleaning & transformation. In practice, there is another Python script that runs all of the other extraction code scripts all at once. In essence, the Python scripts I wrote go through the following stages.

1. Setup
    a. Import the following packages: requests, pandas, time, datetime, dateutil, gspread.
    b. Create function to make request to Facebook Graph API.
    c. Create function that paginates through API request output if all output is not visible at once.
    d. Instantiate variables for API version and request URL.

    e. Retrieve current long-lived access token from text file.
2. Import data
    a. Retrieve social media data file which is in CSV format. The file consists of data that is until the last time the extraction code was run which is typically the week prior. It would take the form of a pandas dataframe.
    b. As there is a date column in the data file, the script takes the most recent date in the column (which is in the first row), and then instantiates as a variable to be used for the request made in the forthcoming stage.
    c. The recent date is then converted from datetime format into unix time format
3. Extraction
    a. Using the API request function along with the request URL & API version variables, and the variable holding the recent date, a request is made to Facebook Graph API to retrieve data of the specified metrics in the request. The output is instantiated to a variable and is of a JSON structure.
    b. Create column heading labels that would be used for the dataframe for the eventual dataframe
    c. Create function that is able to extract the date and metric values from a JSON object and appends to a list. There is a nested JSON object per date.
    d. Create function that applies the aforementioned function throughout the request output per nested JSON object in the form of a loop.
    e. Instantiate the aforementioned function to variable that would be used as the basis to create a dataframe of the newly extracted data.
4. Transformation
    a. The newly extracted data and column labels (created in the prior stage) are used to create a dataframe. The dataframe rows would be in descending order by date (i.e., the most latest date to earliest date).
    b. The dataframes of the imported data and newly extracted data are then combined together.
    c. Some data cleaning is performed to the combined dataframe such as dropping duplicates.
5. Loading
    a. The combined dataframe is then saved as a CSV file. In turn, overwriting the earlier imported social media data file.
    b. Then, the newly saved social media data file's contents are saved to the corresponding Google Sheets file on the organization's Google Drive. Specifically, the existing data in the Google Sheets file is deleted, and then the data in the CSV file are then copied and pasted in the Google Sheets file. This is accomplished through a function that takes parameters consisting of Google Sheets file name, sheet name, and spreadsheet ID, along with authentication via the Google service account private key.

Below is a list of resources that were imperative to setting up an environment to retrieve social media data of the organization's Facebook and Instagram pages.

- Setting up Facebook for Developers and working with Graph Explorer

    - https://www.youtube.com/watch?v=LmhjVT9gIwk&list=PLhpgLgFy42uVqkUa_5P0HZIg0dwbVm96D&index=1
    - https://www.klipfolio.com/blog/facebook-graph-api-explorer
- Facebook Graph API access tokens
    - https://www.igorkromin.net/index.php/2020/10/02/facebook-access-token-error-validating-access-token-session-has-expired/
    - https://www.sociablekit.com/get-facebook-long-lived-user-access-token/
- Wrangling data from Facebook Graph API
    - https://bubbletao.com/2017/05/31/python-in-digital-analytics-automating-facebook-metric-reports/
- Metrics
    - https://supermetrics.com/api/getFields?ds=FB&fieldType=met
    - https://developers.facebook.com/docs/graph-api/reference/v11.0/insights#post_impressions
    - https://prezi.com/acd742tobfjg/quotnamequot-quotpage_fans_countryquot/
    - https://developers.facebook.com/docs/instagram-api/reference/ig-media/insights/
- Requests to Facebook Graph API and Python
    - https://stackoverflow.com/questions/37419788/difference-in-engaged-users-facebook-page-and-engaged-users-insights-api
    - https://stackoverflow.com/questions/38924707/get-post-insights-for-multiple-posts-in-one-call-using-graph-api-2-7/39104504
    - https://stackoverflow.com/questions/44491319/facebook-page-insights-api-to-retrieve-different-insight-metrics-using-python
    - https://stackoverflow.com/questions/38924707/get-post-insights-for-multiple-posts-in-one-call-using-graph-api-2-7/39104504
    - https://stackoverflow.com/questions/7227498/list-of-countries-and-cities-to-be-used-in-facebook-graph-api-for-targeting
- The full analysis process
    - https://medium.com/analytics-vidhya/analyse-your-personal-facebook-data-with-python-5d877e556692
    - https://www.youtube.com/playlist?list=PLhpgLgFy42uVqkUa_5P0HZIg0dwbVm96D
- Setting up Google service account via Google Cloud Console to use Python and Google Sheets
    - https://www.youtube.com/watch?v=cnPlKLEGR7E&list=LL&index=11
    - https://www.youtube.com/watch?v=bu5wXjz2KvU&list=LL&index=10
    - https://www.youtube.com/watch?v=TNIoGW8NzrY&list=LL&index=9
    - https://www.youtube.com/watch?v=2yIcNYzfzPw&list=LL&index=7
    - https://medium.com/craftsmenltd/from-csv-to-google-sheet-using-python-ef097cb014f9
- Specific ways to interact with Google Sheets using Python & gspread package
    - https://github.com/burnash/gspread
    - https://docs.gspread.org/en/latest/user-guide.html#updating-cells

- https://github.com/PrettyPrinted/youtube_video_code/tree/master/2021/10/14/How%20to%20Use%20Google%20Sheets%20With%20Python%20(2021)
- https://practicaldatascience.co.uk/data-science/how-to-read-google-sheets-data-in-pandas-with-gspread
- https://readthedocs.org/projects/gspread-pandas/downloads/pdf/latest/

Environment Setup for Dashboard Application

The dashboard application's overall structure was stated in an earlier section of this report. In this section, the dependencies to ultimately create the dashboard application will be flushed out more concisely. This section would also describe the required project structure that enables a multi-page Dash application to run sucessfully. Starting out, below is a list that contains all dependencies related to environment setup at this current stage of the project.

Necessary dependencies to begin constructing a multi-page Dash application

- Technology required
  - Computer with a Windows or Mac OS X operating system
- Knowledge required
  - Basic understanding of object oriented programming in regards to instantiating variables, creating & applying functions, creating classes
    - Knowledge & holistic understanding gained from the following Douglas College courses
      - CSIS 1175: Introduction to Programming I
      - CSIS 2175: Advanced Integrated Software Development
  - Understanding of software development in creating multi-layer applications
    - Knowledge & holistic understanding gained from the following Douglas College courses
      - CSIS 2300: Database I
      - CSIS 3300: Database II
      - CSIS 3175: Mobile Application Development I
      - CSIS 3275: Software Engineering
  - Basic understanding of descriptive statistics (E.g., calculating average)
    - Knowledge & holistic understanding gained from the following Douglas College courses
      - BUSN 2429: Business Statistics I
      - BUSN 3431: Business Statistics II
      - CSIS 3360: Fundamentals of Data Analytics
  - Data storytelling for the purpose of selecting an appropriate type of chart for a given dataset
    - Knowledge & holistic understanding gained from the following Douglas College courses
      - CSIS 3860: Data Visualization
  - Web development principles in terms of knowing various web elements and their respective html tags, as well as basic website design

        - Knowledge & holistic understanding gained from the following Douglas College courses
            - CSIS 1280: Multimedia Web Development
- Software
    - Python programming language version 3.8.9+; installed directly from the Python website or from the Anaconda version
        - Download link for latest Python distribution = https://www.python.org/
        - Download link for Anaconda distribution = https://www.anaconda.com/
        - Python scripts are used to code the project overall
        - Jupyter notebooks are used to demo certain portions of the dashboard application (e.g., data filtering, aggregation, visualization) that would eventually make its way into new/existing Python scripts to contribute to the creation
    - Microsoft VS Code; used as an integrated development environment
        - Download link = https://code.visualstudio.com/
    - Web browsers such as Google Chrome, Brave, or Firefox are used as the medium by which the dashboard application is functionable after running the code
        - Google Chrome download link = https://www.google.com/intl/en_ca/chrome/
        - Brave download link = https://brave.com/
        - Firefox download link = https://www.mozilla.org/en-CA/firefox/
- Additional dependencies (some of them were touched upon in more detail in the previous environment setup section)
    - Zyp Art Gallery Google account that allows access to the organization's Google Drive
    - Google service account private key for the purpose of authentication so that the dashboard application can import the social mediat data files directly from the organization's Google Drive
    - Country code ISOs that are in CSV file format (sources noted in References section of the report)
- Python packages (packages not pre-packaged by the Python are installed via Command Prompt/PowerShell/Terminal via Pip)
    - Gspread package; used to import the social media data files from the organization Google Drive
        - Installation = pip install gspread
    - Pandas package; imported social media data files take the form of dataframes, and are also utilized as the basis of creating visualizations after performing any necessary aggregation and/or filteration
        - Installation = pip install pandas
    - Numpy package; used to perform simple calculations such as finding the average of a given metric value from start date to end date selected
        - Installation = pip install numpy
    - Datetime package: datetime, time, timedelta

        - Installation = pip install DateTime
        - Datetime used for converting string dates to appropriate datetime format; this is also helpful when handling datepickers
        - Timedelta is used to set the default start date to 30 days prior to the default end date
    - Calendar package (imported but not actually used)
    - Plotly package: express, graph objects
        - Installation = pip install plotly
        - Express used to create visualizations such as bar charts, line charts, pie charts, and area charts
        - Graph objects package used to create choropleths and other map related charts
    - Dash package: Dash, dcc, html, Input, Output, State, Callback, Dash table
        - Instllation = pip install dash
        - Dash is used to instantiate the dashboard application so that is run using a local server via a web browser
        - Dcc is used to create web elements such as datepickers, dropdowns and checklists, as well as tabs
        - Html is used to create basic web elements such as headings and paragraphs
        - Input, Output, State, and Callback are used to reflect change on a webpage based on user manipulation of the interactive web elements such as those that serve the intent of filtration
        - Dash table is imported but not used
    - Dash bootstrap elements package; used to create interactive elements such as dropdowns and buttons, as well as webpage structural elements such as containers, rows, columns, and card tiles
        - Installation = pip installation dash-bootstrap-components
    - Sys package; used to import other Python script's variables and functions to be used in another Python script
        - Installation = pip install os-sys
    - GUnicorn package; used for managing deployment of a project existing on a local machine or GitHub to Heroku
        - Installation = pip install gunicorn

<u>General project structure for my version of a multi-page Dash application</u>

A local folder/directory for the project would have to be created and then opened using an IDE or code editor. In my case, I used VS Code. The structure below describes required project structure when running the application on a local server (i.e., on a local machine/computer).

- Project's root directory
    - Apps folder: this folder contains scripts dedicated to rendering the main content of the dashboard application, which are the visualizations of the social media data and interactive web elements to filter/alter the visualizations. Each script corresponds to a dedicated page of the application.

        - An empty “__init__.py” file is needed here as well.
    - Assets folder; this folder contains files such as images, static files, and cross-referencing data files.
        - An empty “__init__.py” file is needed here as well.
    - Data folder: This folder contains data files that are used as the data source of the various visualizations. Data files could be included in the Apps folder, but I decided to put all data files in a separate folder due so many being used in the application.
    - App Python script: This script instantiates the Dash application itself as well as the server too. In my version, the Dash authentication function is instantiated here as well.
    - Index Python script: This file links all of the scripts within the Apps folder together via a navigation bar, so that when running the application, the user would be able to navigate across the various pages. Also, this script calls the variables from the “app” Python script in the root directory. This in turns ties the whole application centrally to the “index” Python script, which makes sense in a traditional sense as the application is technically a web application.

Further additions would have to be made to the project such that it is deployable to Heroku. This is discussed in the forthcoming section.

Environment Setup for Dash Application Deployment to Heroku

Step 1: Signup for a Heroku account

In my case, I signed up for a Heroku account. It was easy to make an account as I simply had to follow the instructions on screen. As the application I am creating is for an organization, I created the Heroku account using my organization email. I also included the name of the organization I am a part of, which is Zyp Art Gallery, as well as the primary development language, which is Python. Below is the Heroku signup link I used.
https://signup.heroku.com/

Step 2: Install GitHub CLI tools & Heroku CLI tools

Installation of GitHub command line tools (CLI tools) and Heroku command line tools (CLI tools) is important in being able to sucessfully deploy the application to Heroku. Heroku CLI tools is required to authenticate oneself via terminal. Both Heroku CLI tools and GitHub CLI tools are used to to create a Heroku app, which is technically a GitHub repository that is masked behind Heroku. GitHub CLI tools are used to add & commit the files within the project directory. Then, both Heroku CLI tools & GitHub CLI tools are used to push the application to Heroku. In other words, push to the application to the GitHub repository that is masked behind Heroku. Below are the download links. I personally installed GitHub CLI tools first, and then installed Heroku CLI tools.

- GitHub CLI tools: https://git-scm.com/downloads
- Heroku CLI tools: https://devcenter.heroku.com/articles/heroku-cli

Step 3: Install PyCharm Community edition

Even though I used VS Code to develop the application, the Pycharm Community Edition IDE instantly creates a virtual envrionment upon creation of a new project. In turn, once the application is created, a requirements text file containing only the dependencies necessary to run the application would be present; not all Python dependencies that are installed on your local machine. There are more involved ways to do this, but the PyCharm IDE makes this all very quick and easy. Below is the download link to the PyCharm Community Edition IDE, as well as the YouTube video link I followed to install the IDE on my computer.

- Download link: https://www.jetbrains.com/pycharm/download/#section=linux
- YouTube video showing how to install PyCharm on Windows 11: https://www.youtube.com/watch?v=2RSDlwWH0XI&list=LL&index=5

Step 4: Create Heroku version of the project

The dashboard application is already in an optimal structure to run on a local server. Certain adjustments need to be made such that it is in an optimal structure to be deployed on Heroku. Below are a list of sequential sub-steps that I followed.

1. Firstly, a new Python project directory should be created via the newly installed PyCharm IDE. This newly created project directory would eventually become the Heroku version of the dashboard application.
2. Now, one must copy the files of the local server version of the project, and then paste it into the project directory of the Heroku version. I used the Windows File Explorer to manually do this but one can also do this via command line or terminal (unix) commands.
3. Make sure that the virtual environment is active whilst having Heroku version of the project open via PyCharm. A way to check is to open the terminal within PyCharm, and then check to see if "(venv)" appears before the current working directory's file path (i.e., project's root directory file path). To access the terminal, simply select the terminal tab on the bottom-left bar of the PyCharm window.
4. Run the following command in the terminal. This command is meant to initialise an empty git repository that would be used to house the application within Heroku. Although from my practice, this was not the most necessary step and could be skipped if no errors occur in forthcoming steps. It is still safe to perform this step.

```
git init
```

5. Install the gunicorn Python package to the virtual environment. This packages helps enable deployment to Heroku.

```
pip install gunicorn
```

6. Create a new file in the root directory of the project called ".gitignore". Then write the following in the newly created file.

```
venv
*.pyc
.DS_Store
```

```
.env
```

7. Create another new file in the root directory of the project called “Procfile”. Then write the following in the newly created file. Notice below that “index” is included in the snippet below. This is indicating to gunicorn that the “index” Python script is the central component of the application. Thus, the webpage that this script renders is the first one the user sees when they access the application.

```
web: gunicorn index:server
```

8. As the terminal is already open, install the most important Python packages to the virtual environment that are necessary to run the application. This is accomplished by using the ‘pip install’ command in the terminal. Below is a list of the most important packages that served as the primary dependency, as well as installed the related secondary dependencies automatically. Below is the precise command used, as well as a table of dependencies that were the most important along with its version numbers. A full list of the dependencies can be found in the “requirements.txt” file that is available to view on the project’s GitHub repository. More about the “requirements.txt” file is discussed in the next sub-step.

```
pip install <name of package>==<version number>
```

Package Name	Version Number
dash	2.6.0
dash-auth	2.6.0
dash-bootstrap-components	1.0.2
dash-core-components	2.0.0
dash-html-components	2.0.0
dash-table	5.0.0
Flask	2.1.3
gspread	5.4.0
gunicorn	20.0.4
numpy	1.22.0
pandas	1.3.5
plotly	5.5.0

9. Create a “requirements.txt” file that contains all of the required dependencies to sucessuflly run the application. Due to the virtual environment, dependencies that are not relevent to theapplication would not be populated in the “requirements.txt” file. This is accomplished using a pip command via the terminal. The following command will create the “requirements.txt” file in the root directory of the project.

```
pip freeze > requirements.txt
```

10. Then, the Heroki CLI & GitHub commands are triggered. This is to deploy the application to Heroku. Specifically, I applied certain commands in the terminal that sequentially performed the following.
    a. Logged into Heroku
        i. The terminal would hint that you press a key on the keyboard, other than “q”, to be redirected to a login-authentication page on your default web browser.
        ii. One must simply login with their Heroku username and password in the login-authentication page. Then, close the web browser entirely so that Heroku CLI tools knows that you are logged in already.
    b. Created a Heroku app that corresponds with the name of the project, which creates a git repo for Heroku. Below is the command.

```
heroku create <name of app>
```

    c. Performed git commands to add & commit project files.

```
git add .
git commit -m "<message>"
```

    d. Pushed the project to the master branch of the git repo.
    e. Scale the web app to use 1 dyno (free tier)

    Below is a code snippet of what the commands precisely were in my case.

```
heroku login

heroku create zyp-social-media-dash-app

git add .

git commit -m "Initial Attempt"

git push heroku master

heroku ps:scale web=1
```

11. The application should be successfully deployed now. It is now a Heroku application deployed to the internet. The application may take some time to load after some idle time as the Heroku app goes to sleep. The application would be instantly accessible after just

deploying it. Below is the structure of the URL link of a Heroku app, and how to access the Heroku app for this project.

- General URL of Heroku app: “<name of app>.herokuapp.com”
    - The Heroku app link for this project: https://zyp-social-media-dash-app.herokuapp.com/

The deployment aspect of this project was quite challenging. I found it the most informative to refer to the links below, and follow along.

- https://towardsdatascience.com/beginners-guide-to-building-a-multi-page-dashboard-using-dash-5d06dbfc7599
- https://www.youtube.com/watch?v=j3VvVaNnDH4
- https://www.youtube.com/watch?v=RMBSQ6leonU&list=PLh3I780jNsiShH-OYSfosz2x1KzRm_F_U&index=3
- https://www.youtube.com/watch?v=Gv910_b5ID0&list=PLh3I780jNsiShH-OYSfosz2x1KzRm_F_U&index=7
- The links below served as reference and assisted to rectify certain errors
    - https://towardsdatascience.com/deploying-your-dash-app-to-heroku-the-magical-guide-39bd6a0c586c
    - https://stackoverflow.com/questions/61452429/plotly-dash-app-gives-error-after-deployment-to-heroku
    - https://stackoverflow.com/questions/24114676/git-error-failed-to-push-some-refs-to-remote
    - https://stackoverflow.com/questions/50026190/heroku-fails-to-install-pywin32-library
    - https://stackoverflow.com/questions/63880206/how-to-run-bootstrap-css-components-in-heroku-python-dash-application
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/tree/master/Deploy_App_to_Web/Authentication
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/tree/master/Deploy_App_to_Web/Multipage_App

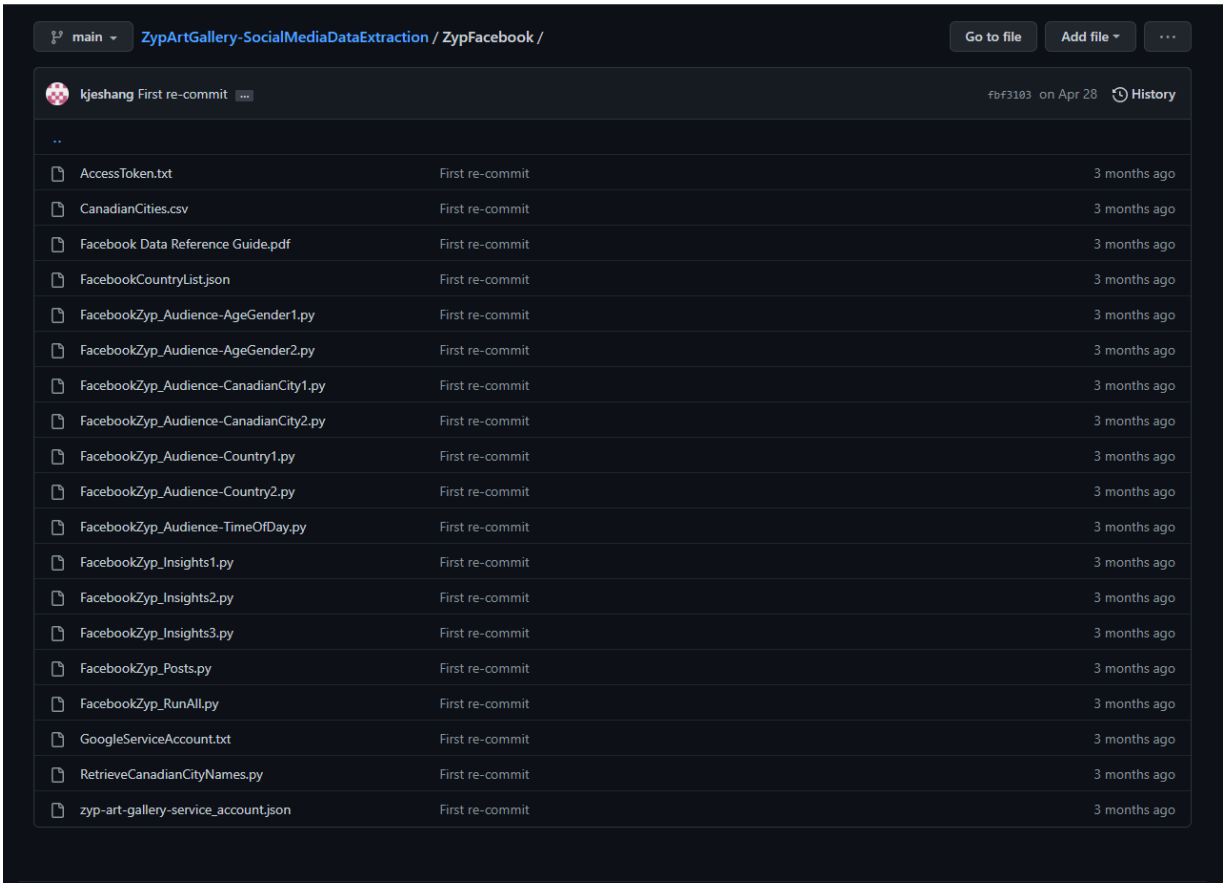
Deploy Changes to Dash Application via Heroku

As the data source for the application is using CSV files of the social media data. On a periodic basis, the application would have to updated to include the most recent social media data files. Deployment of changes could ideally be performed every week or every two weeks so that the application’s visualizations reflect recency of social media insights. Also, to avoid the Heroku application from timing out due to large data processing of the CSVs, I will remove data that of older years and quarters. At the time of this writing, the dataset used for the Facebook Canadian City section consists of row values of the current year quarter only. As the application has already been deployed with the help of PyCharm, when it comes to deployment of changes, I simply used VS Code. Below is a list of steps I followed.

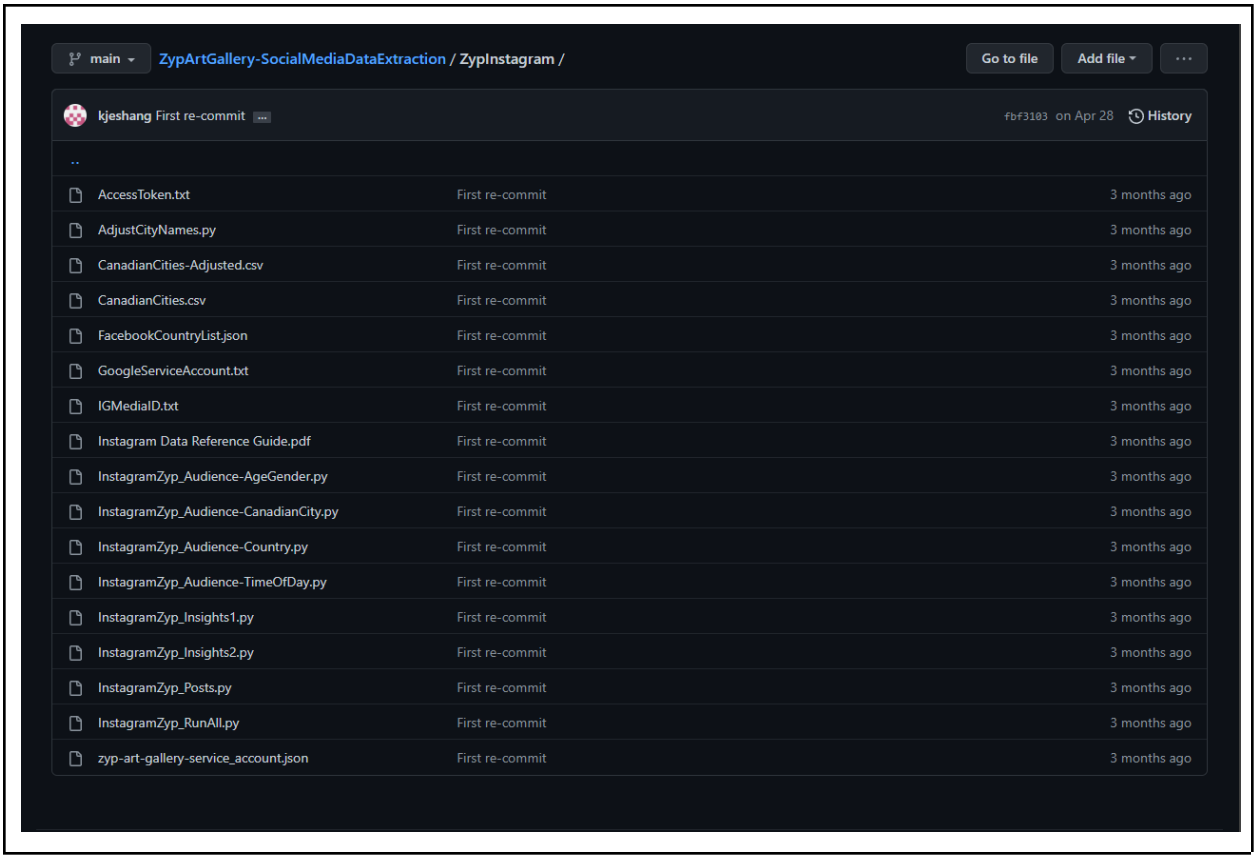
Step 1: Run Social Media Extraction code

This step is technically out-of-scope of the project, but a necessary one in order to retrieve up-to-date social media data as CSV files. Below is the link to the social media extraction code from my personal GitHub. It is a not a working version as it requires the Facebook Graph API access token, as well as referential screenshots. I have a working version on my local computer that contains a folder within the respective directories of both Facebook and Instagram extraction scripts. This folder houses the actual social media data files.
https://github.com/kjeshang/ZypArtGallery-SocialMediaDataExtraction

Below is a screenshot of the Facebok social media extraction code files from my personal. There are various extraction scripts. All I need to do is to run the “ZypFacebook_RunAll” Python script to run all other data extraction scripts.



Below is a screenshot of the Instagram social media extraction code files from my personal. There are various extraction scripts. All I need to do is to run the “ZypInstagram_RunAll” Python script to run all other data extraction scripts.



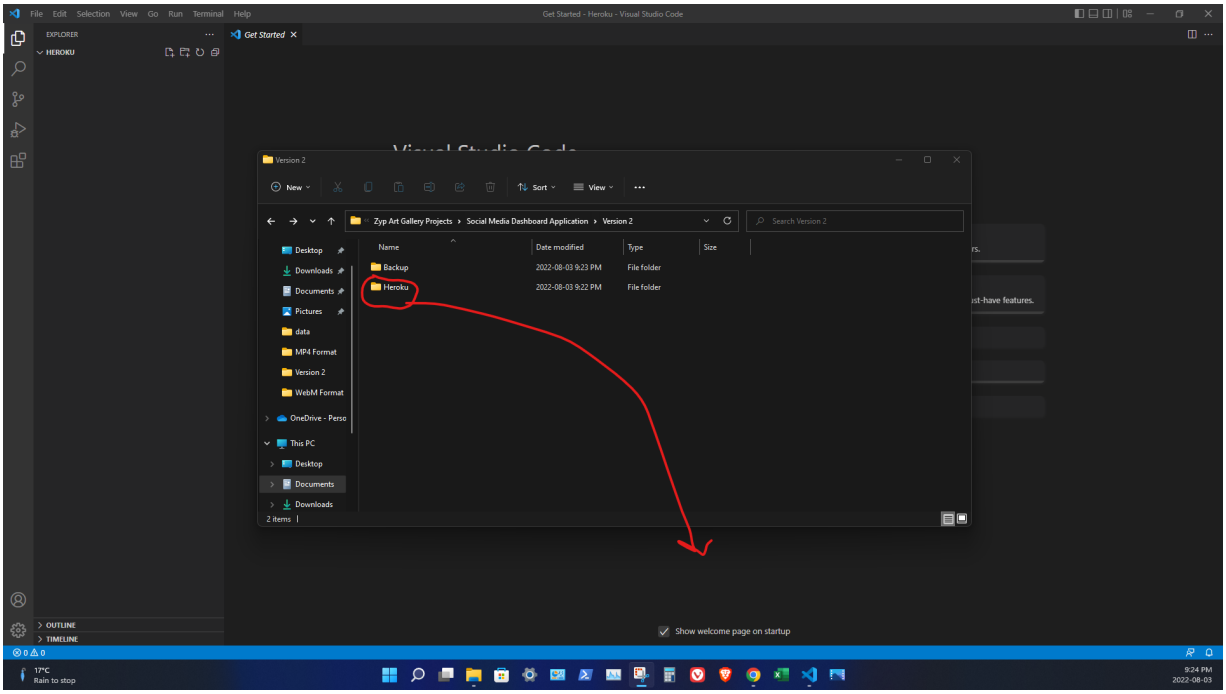
Step 2: Clone the application's Heroku repository to the local machine

This step would be communicated via a sequential list of sub-steps. There are also referential screenshots that serve as a visual aid.

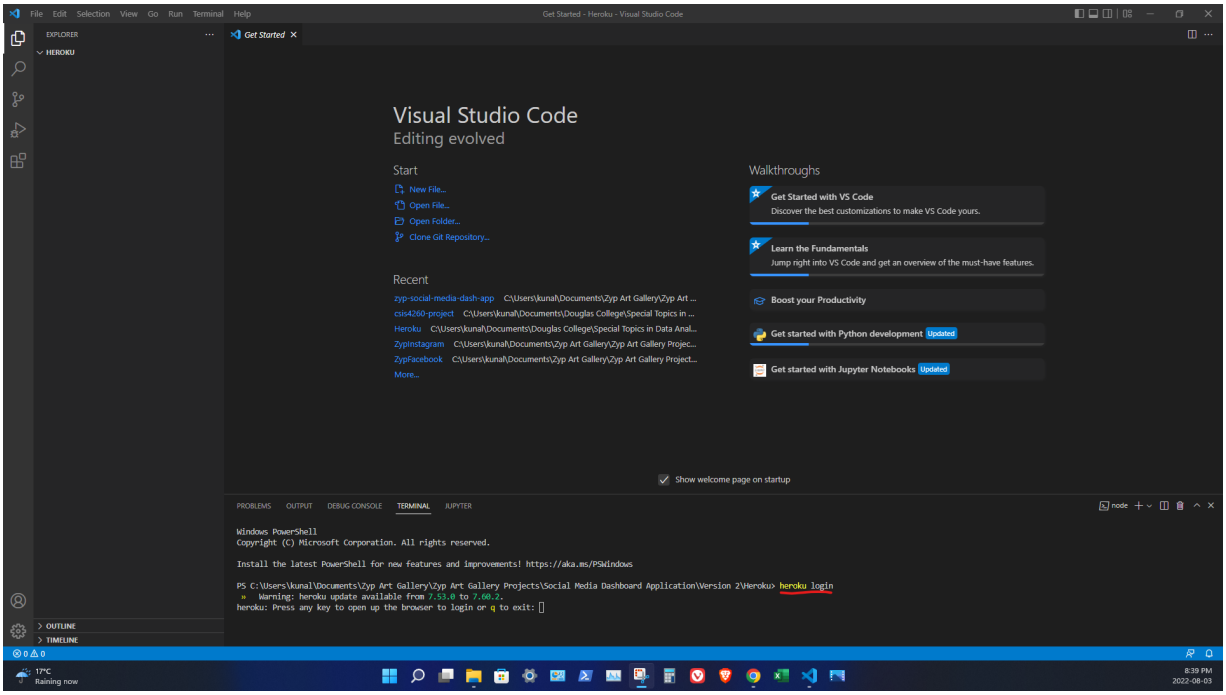
1. Create a new folder in any parent directory called "Heroku" using File Explorer.
2. Launch VS Code. Then, open the "Heroku" folder directory within VS Code.
3. Open the terminal and check to see that the current working directory is the recently opened "Heroku" folder. Within terminal, log into Heroku via Heroku CLI tools, and then clone the Heroku respository of the application to the local machine. After that, change the current working directory so that it is that of the recently cloned application. Below are the precise commands I used in the terminal.

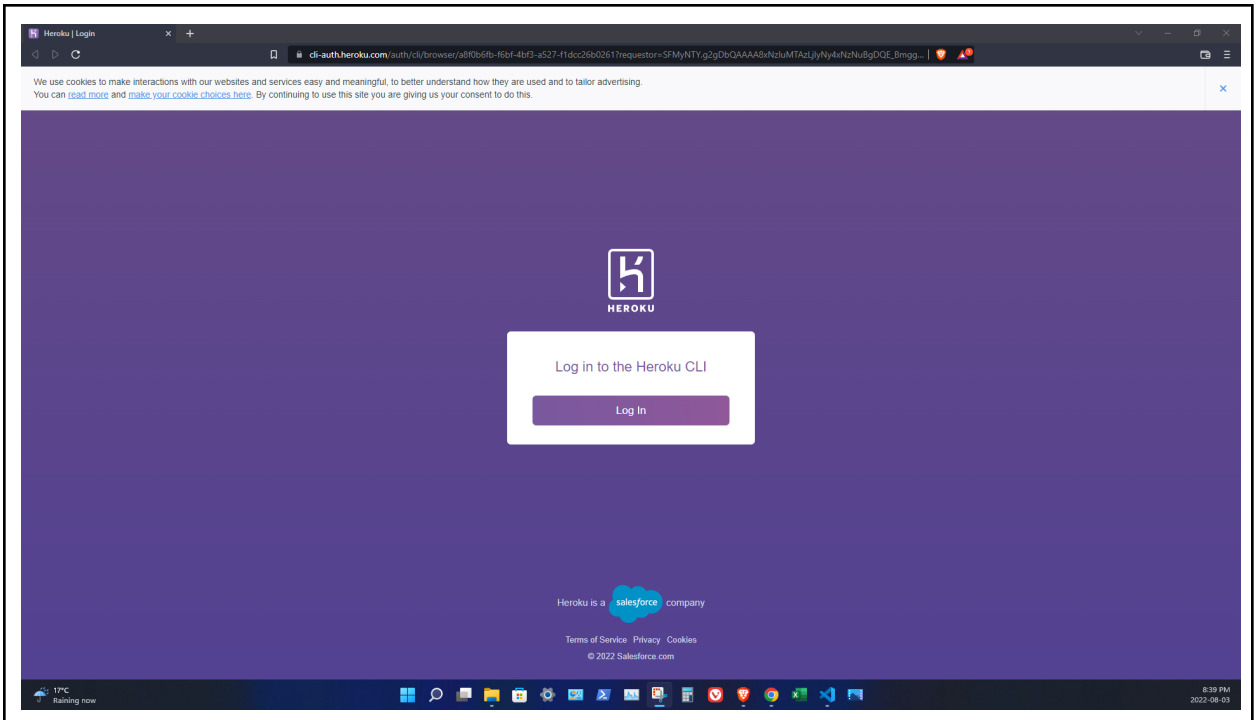
```
$ heroku login
$ heroku git:clone -a zyp-social-media-dash-app
$ cd zyp-social-media-dash-app
```

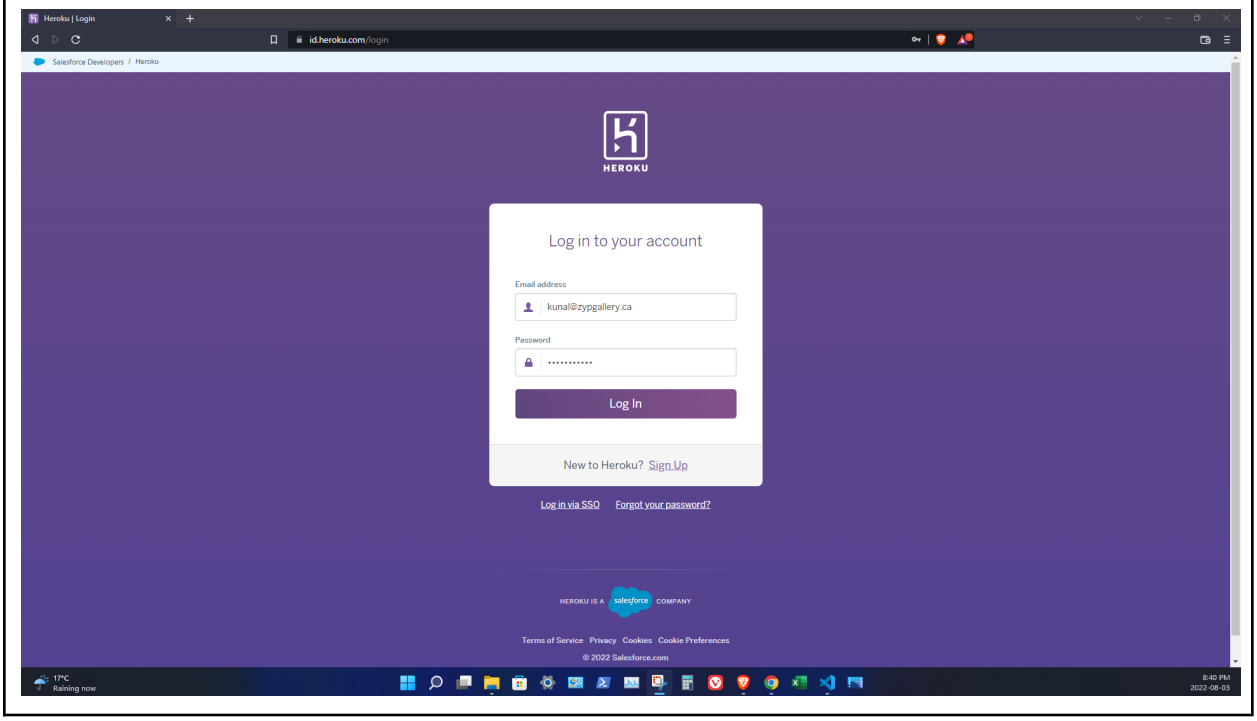
Create new folder called "Heroku" in a parent directory using using File Explorer. Launch VS Code. Drag the folder to the VS Code window to open it.

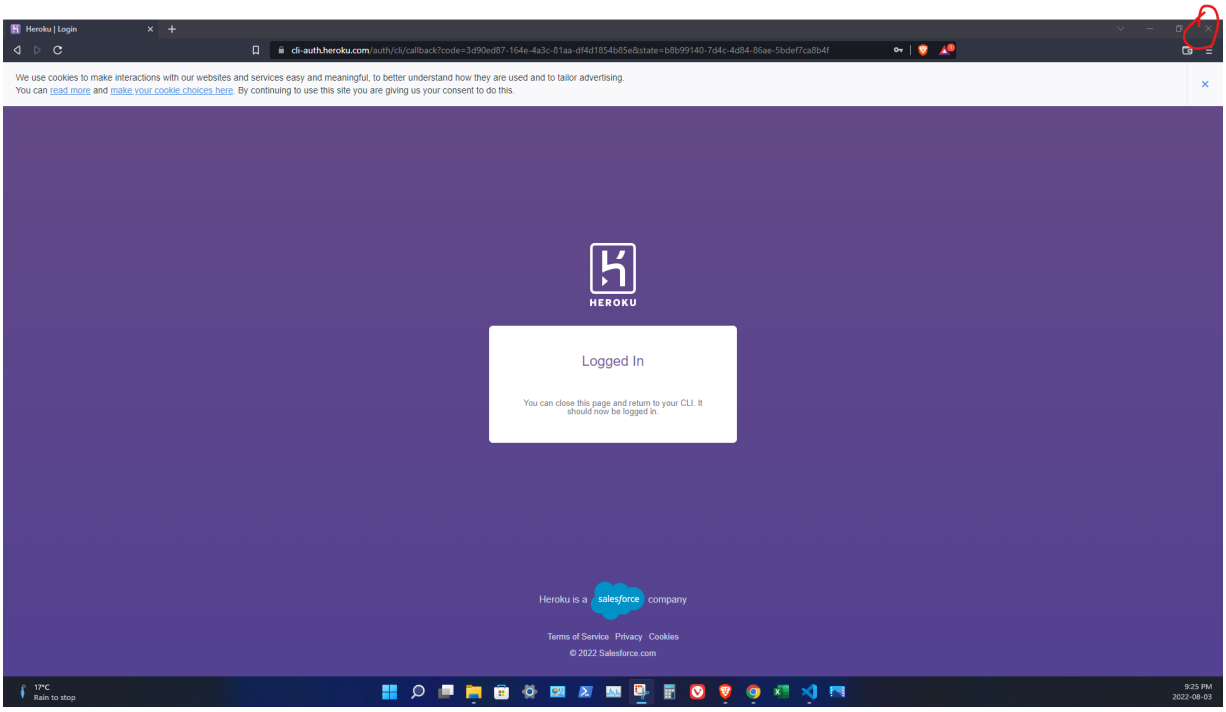


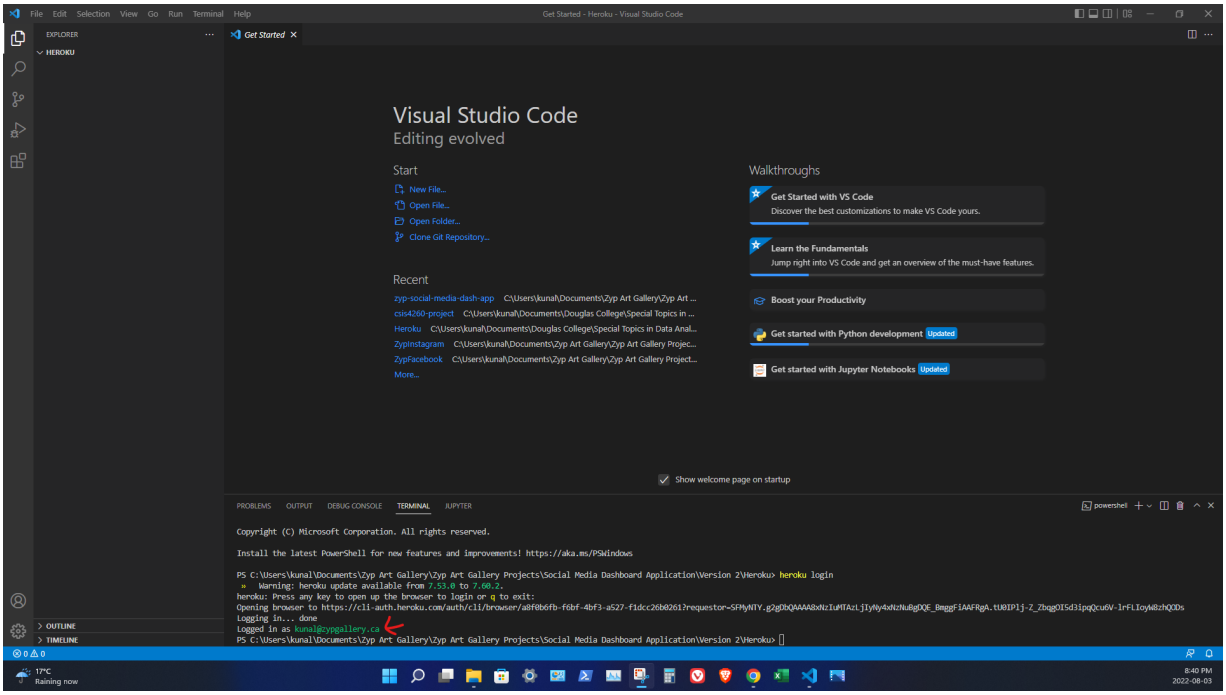
Launch the terminal. Perform the required commands to log into Heroku. Successful Authentication would require entering username and password in webpage that pops up on the default browser. Then one would have to close the webpage to complete authentication.



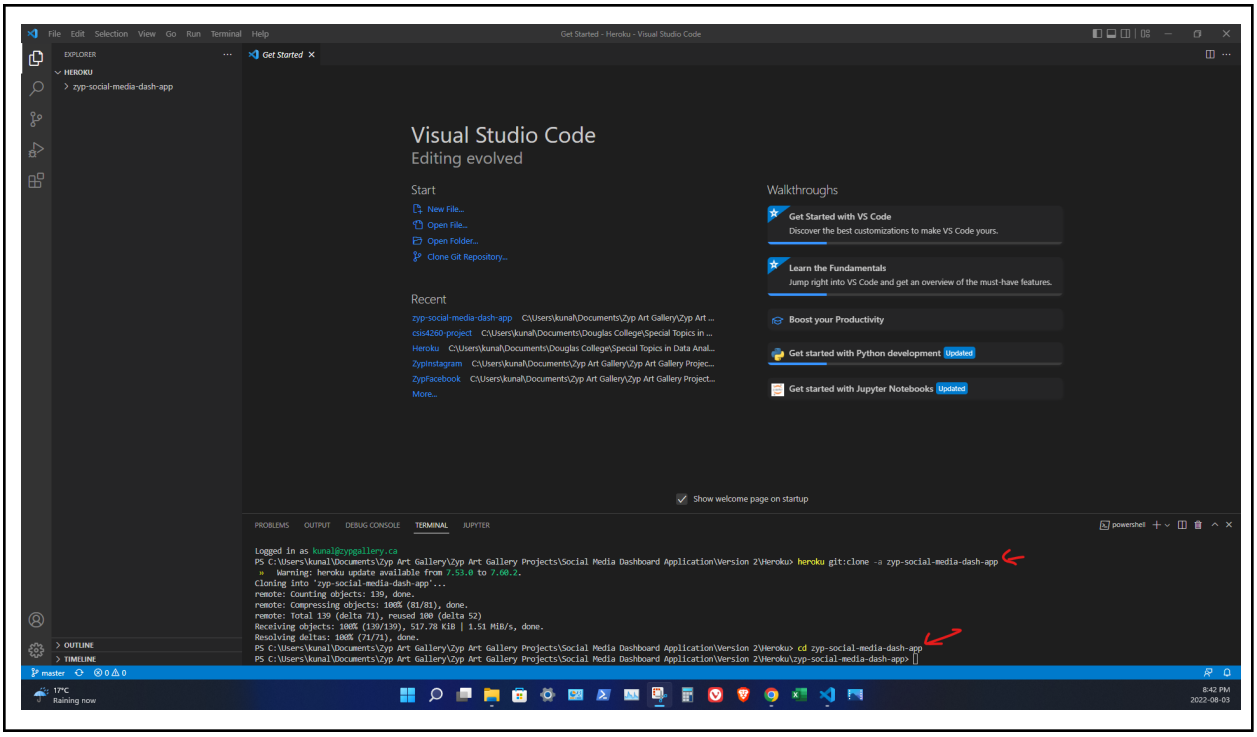

Heroku | Login
cli-auth.heroku.com/auth/cli/browser/a8f0b6fb-f6bf-4bf3-a527-f1dcc26b0261?requestor=SFMyNTY.g2gDbQAAAA8xNzIuMTAzLjIyNy4xNzNuBgDQE_8mgg...
We use cookies to make interactions with our websites and services easy and meaningful, to better understand how they are used and to tailor advertising.
You can read more and make your cookie choices here. By continuing to use this site you are giving us your consent to do this.
HEROKU
Log in to the Heroku CLI
Log In
Heroku is a salesforce company
Terms of Service Privacy Cookies
© 2022 Salesforce.com
17°C
Raining now
8:39 PM
2022-08-03


Heroku | Login
id.heroku.com/login
Salesforce Developers / Heroku
HEROKU
Log in to your account
Email address
kunal@zypgallery.ca
Password
Log In
New to Heroku? Sign Up
Log in via SSO
Forgot your password?
HEROKU IS A salesforce COMPANY
Terms of Service Privacy Cookies Cookie Preferences
© 2022 Salesforce.com
17°C
Raining now
8:40 PM
2022-08-03





Now the application from Heroku can be cloned to the local machine. Then, the current directory must be switched to that of the application itself.



Step 3: Copy the social media data files to the cloned Heroku application's directory

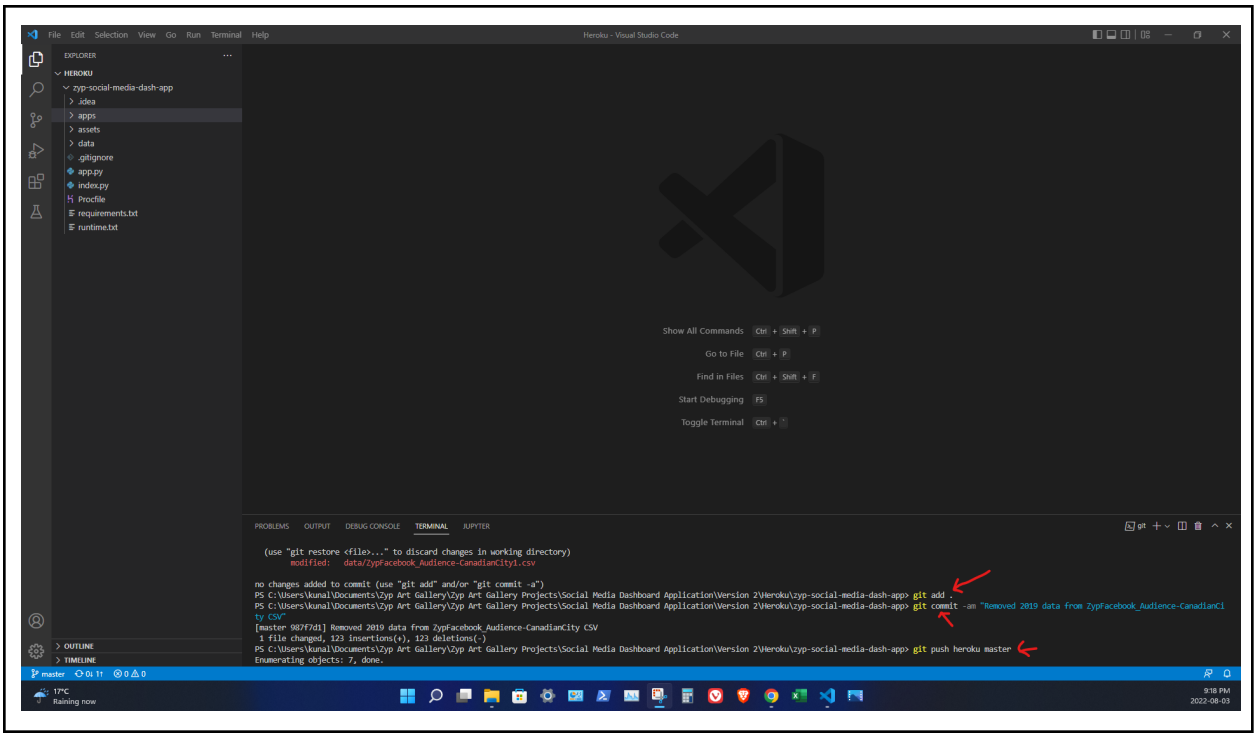
This step is quite self explanatory. Simply copy the social media data CSV files from the data folders of the Facebook and Instagram directories respectively, and paste it in the data folder of the cloned Heroku project's directory.

Step 4: Deploy to the changes made to Heroku

After the 'newly extracted' social media data files are copied to the cloned Heroku application's data folder, the changes made can be deployed to Heroku itself. This can be accomplished using commands enacted within the terminal. Below are the precise commands I used, as well as referential screenshots that serve as a visual aid.

```
$ git add .
$ git commit -am "Change deployment message"
$ git push heroku master
```

Commands that enact deployment after changes are made to the Heroku version of the application on the local machine.



Step 5: Access the application online via Heroku to see deployed changes

All of the aforementioned steps have been completed. Now the application can be accessed again, and the changes would be reflected in the data visualization and the interactive web elements as they would reflect more recent dates. Below is the link to the application on Heroku as well as the terminal output from steps 1 to 5.

https://zyp-social-media-dash-app.herokuapp.com/

Terminal Output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\kunal\Documents\Zyp Art Gallery\Zyp Art Gallery Projects\Social
Media Dashboard Application\Version 2\Heroku> heroku login
 »   Warning: heroku update available from 7.53.0 to 7.60.2.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to
https://cli-auth.heroku.com/auth/cli/browser/a8f0b6fb-f6bf-4bf3-a527-f1dcc2
6b0261?requestor=SFMyNTY.g2gDbQAAAA8xNzIuMTAzLjIyNy4xNzNuBgDQE_BmggFiAAFRgA
.tU0IPlj-Z_ZbqgOI5d3ipqQcu6V-lrFLIoyW8zhQODs
Logging in... done
```

```
Logged in as kunal@zypgallery.ca
 »   Warning: heroku update available from 7.53.0 to 7.60.2.
Cloning into 'zyp-social-media-dash-app'...
remote: Counting objects: 139, done.
remote: Compressing objects: 100% (81/81), done.
remote: Total 139 (delta 71), reused 100 (delta 52)
Receiving objects: 100% (139/139), 517.78 KiB | 1.51 MiB/s, done.
Resolving deltas: 100% (71/71), done.
PS C:\Users\kunal\Documents\Zyp Art Gallery\Zyp Art Gallery Projects\Social
Media Dashboard Application\Version 2\Heroku> cd
zyp-social-media-dash-appPS C:\Users\kunal\Documents\Zyp Art Gallery\Zyp
Art Gallery Projects\Social Media Dashboard Application\Version
2\Heroku\zyp-social-media-dash-app>


git status
On branch master
Your branch is up to date with 'heroku/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   data/ZypFacebook_Audience-CanadianCity1.csv

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\kunal\Documents\Zyp Art Gallery\Zyp Art Gallery Projects\Social
Media Dashboard Application\Version 2\Heroku\zyp-social-media-dash-app> git
add .
PS C:\Users\kunal\Documents\Zyp Art Gallery\Zyp Art Gallery Projects\Social
Media Dashboard Application\Version 2\Heroku\zyp-social-media-dash-app> git
commit -am "Removed 2019 data from ZypFacebook_Audience-CanadianCity CSV"
[master 987f7d1] Removed 2019 data from ZypFacebook_Audience-CanadianCity
CSV
 1 file changed, 123 insertions(+), 123 deletions(-)
PS C:\Users\kunal\Documents\Zyp Art Gallery\Zyp Art Gallery Projects\Social
Media Dashboard Application\Version 2\Heroku\zyp-social-media-dash-app> git
push heroku master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 5.34 KiB | 15.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0
```

```
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> Using Python version specified in runtime.txt
remote:  !     Python has released a security update! Please consider
upgrading to python-3.10.6
remote:        Learn More:
https://devcenter.heroku.com/articles/python-runtimes
remote: -----> No change in requirements detected, installing from cache
remote: -----> Using cached install of python-3.10.1
remote: -----> Installing pip 22.1.2, setuptools 60.10.0 and wheel 0.37.1
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote: -----> Discovering process types
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 136M
remote: -----> Launching...
remote:        Released v5
remote:        https://zyp-social-media-dash-app.herokuapp.com/ deployed to
Heroku
remote:
remote: This app is using the Heroku-20 stack, however a newer stack is
available.
remote: To upgrade to Heroku-22, see:
remote: https://devcenter.heroku.com/articles/upgrading-to-the-latest-stack
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/zyp-social-media-dash-app.git
   bb6f6cb..987f7d1  master -> master
PS C:\Users\kunal\Documents\Zyp Art Gallery\Zyp Art Gallery Projects\Social
Media Dashboard Application\Version 2\Heroku\zyp-social-media-dash-app>
```

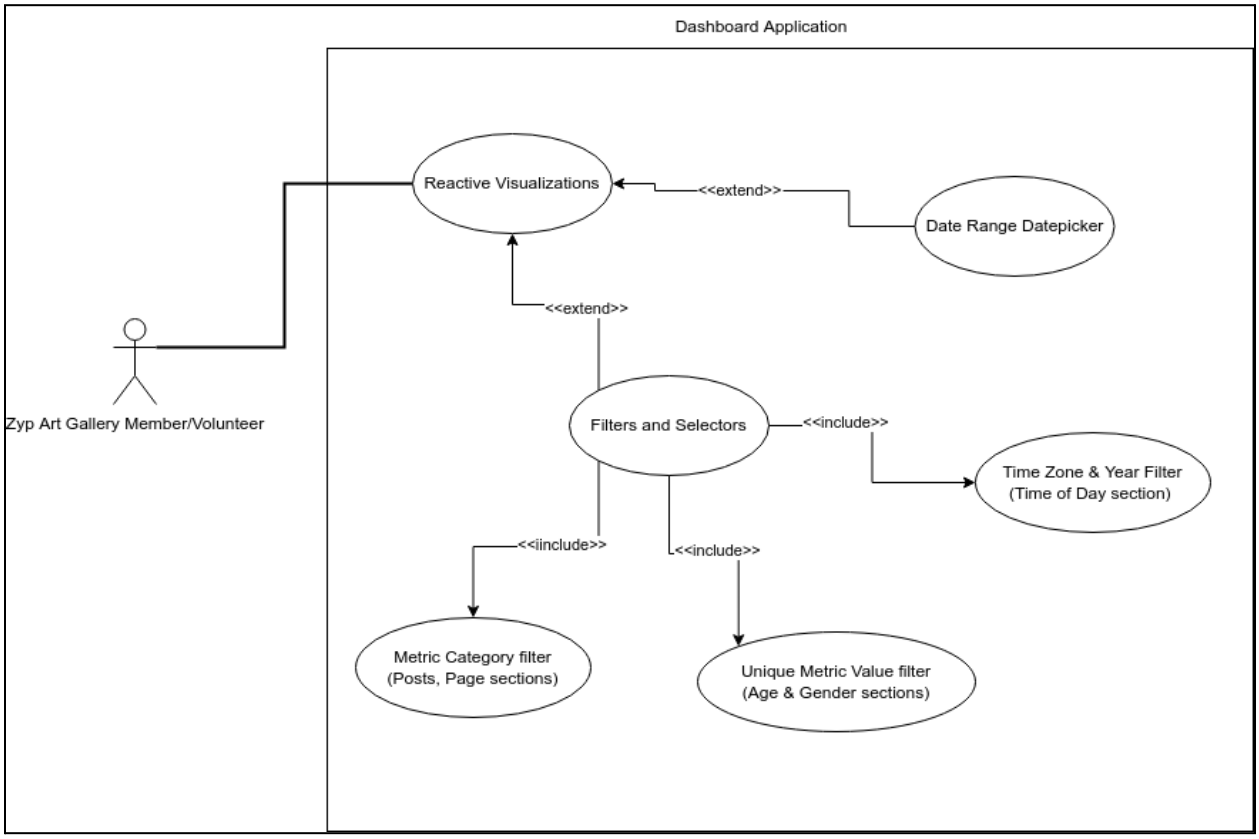
Software Design Architecture

UML

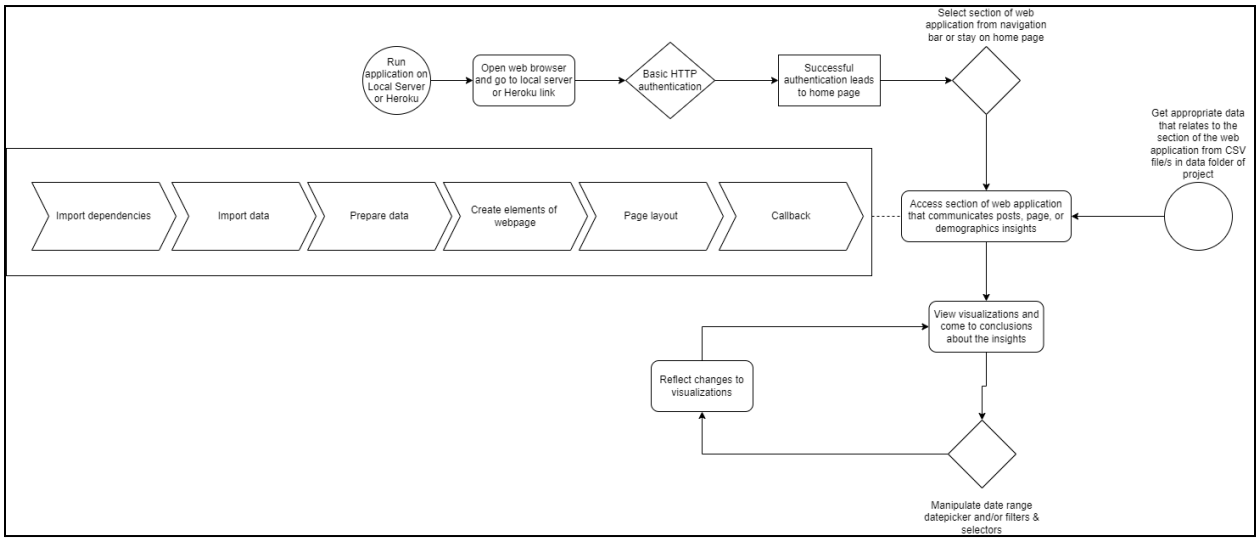
Below is a list of diagrams that could best communicate the software design architecture of the project at its current state in an understandable way. Please note that the UML diagrams were manually created using the free tool “Draw.io”. Thus I apologize for any occurrence of human error.

- Use Case Diagram
- Activity Diagram
- Class Diagram

Use Case Diagram



Activity Diagram

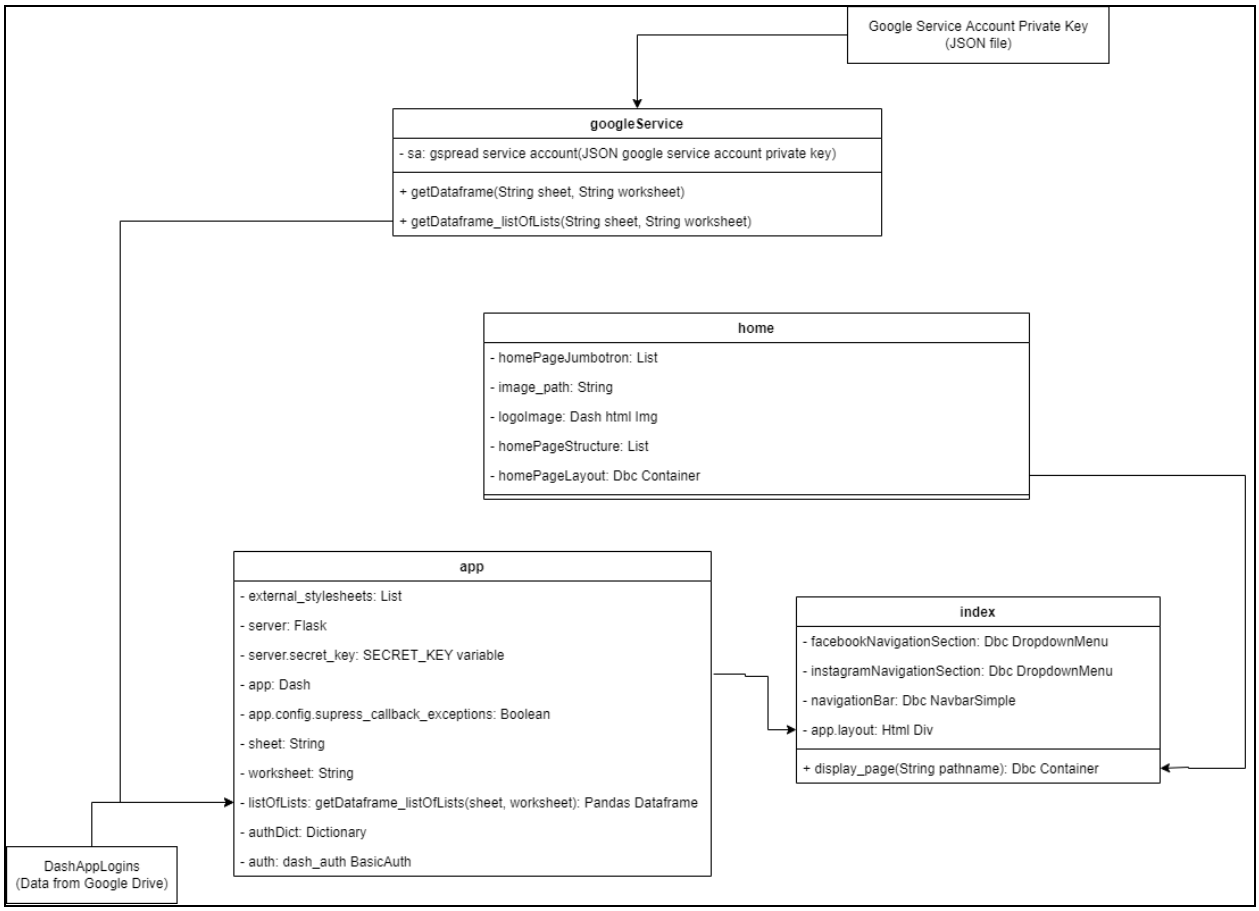


Class Diagrams

It should be noted that Plotly Dash applications typically do not follow the traditional object oriented programmming approach. That being said, I tried my best to create UML class diagrams, that displayed a logical flow, for each Python script file that is dedicated to rendering the Facebook & Instagram sections of the application. The diagrams may not follow the traditional UML class diagram style and format in the case of object oriented programming. Each diagram the flow of the data from the data source to the Python scripts where web elements & visualizations are created using the variables & functions, and then leading eventually to instantiation of the application itself (i.e., leading right upto before the user layer). Below is a brief legend of the UML class diagrams elements as there are many of them.

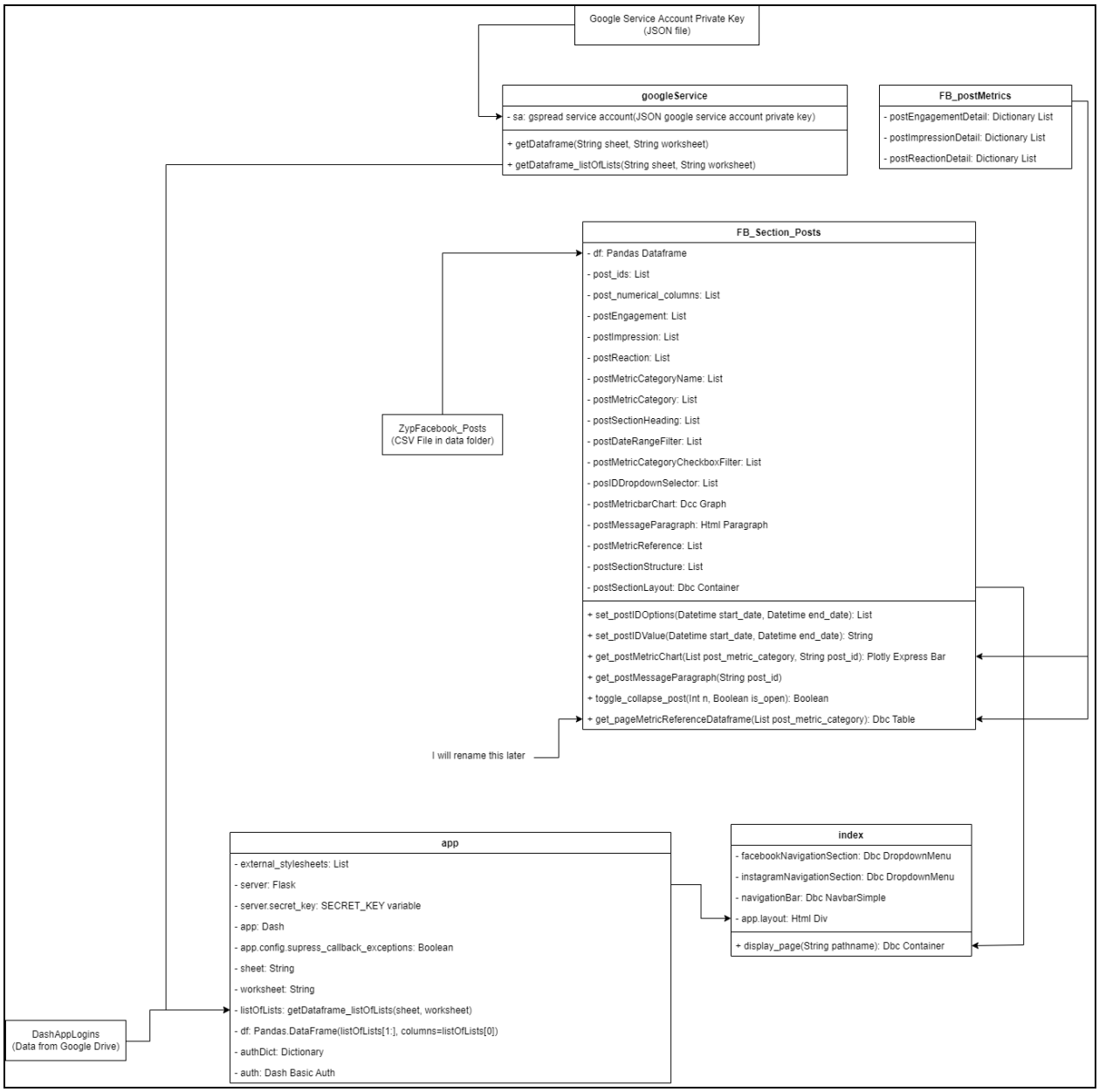
- Box with bold title = Python script
    - Minus sign = Variable
    - Plus sign = Function
- Box with no title = Dependency (JSON or CSV file)
- Arrow = The head of the arrow refers to the Python script/function/variable that is uses the preceding Python script/function/variable

Home page (apps/home.py)

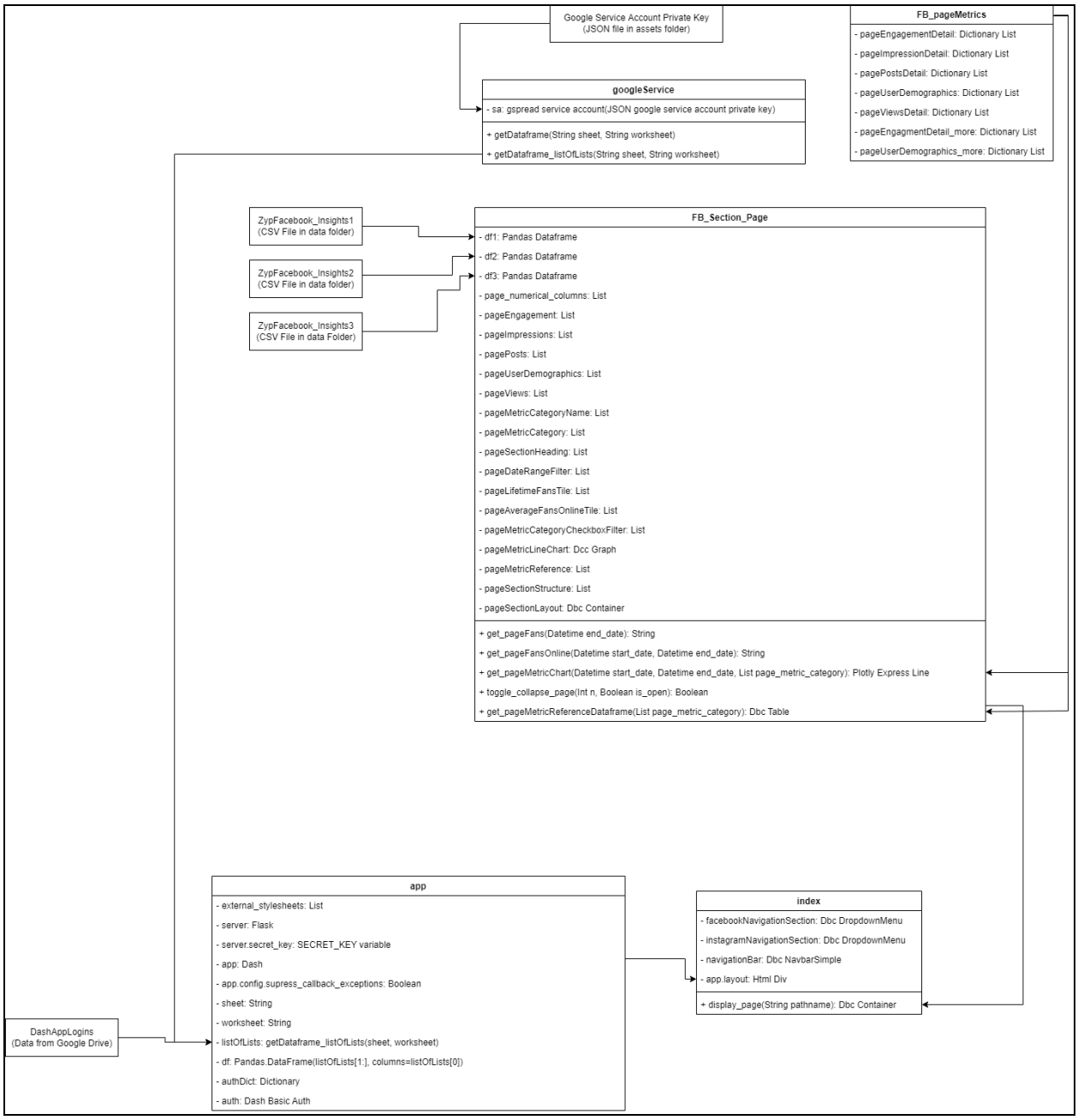


Facebook Sections

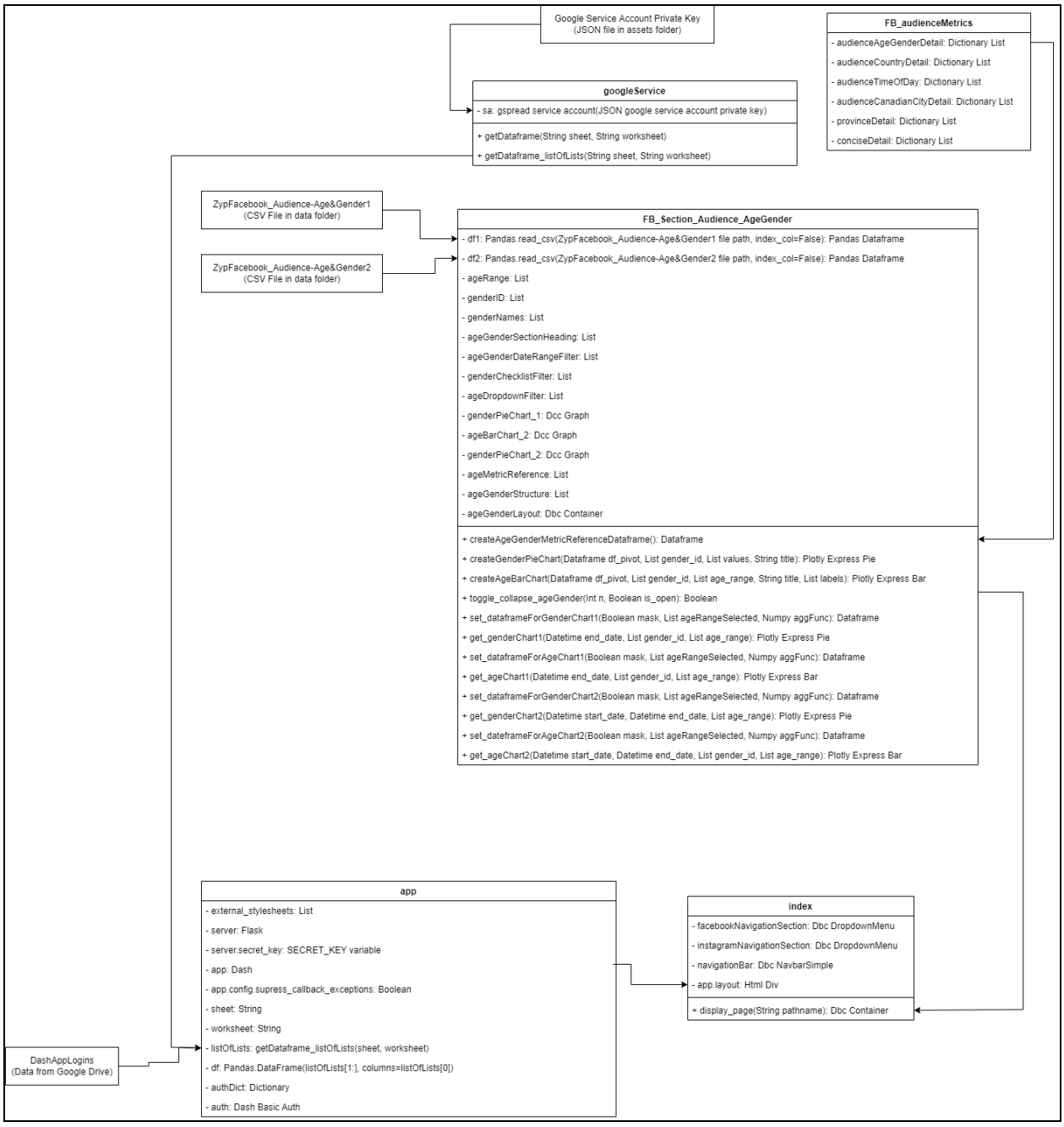
Facebook Post Insights (apps/FB_Section_Posts.py)



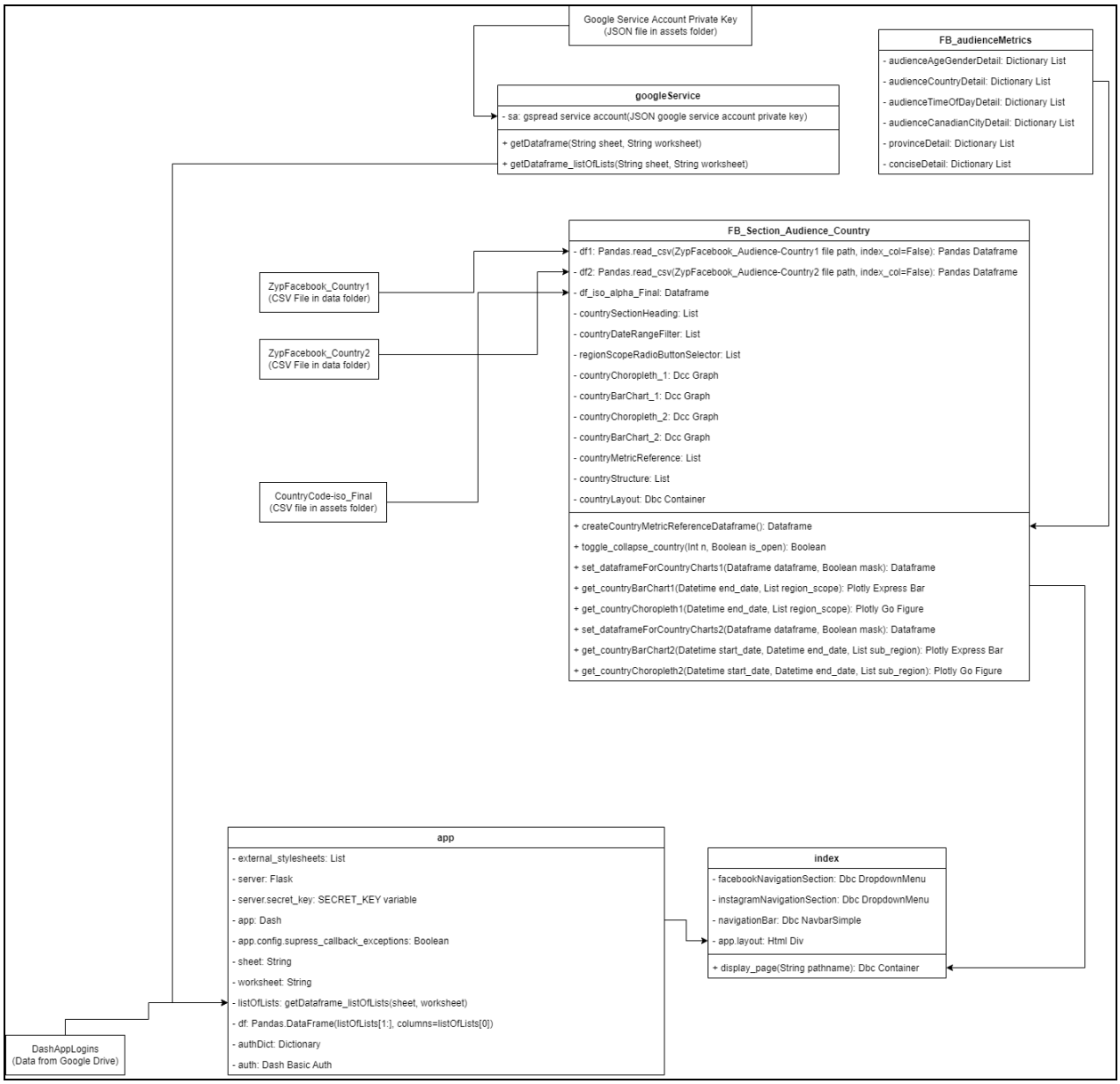
Facebook Page Insights (apps/FB_Section_Page.py)



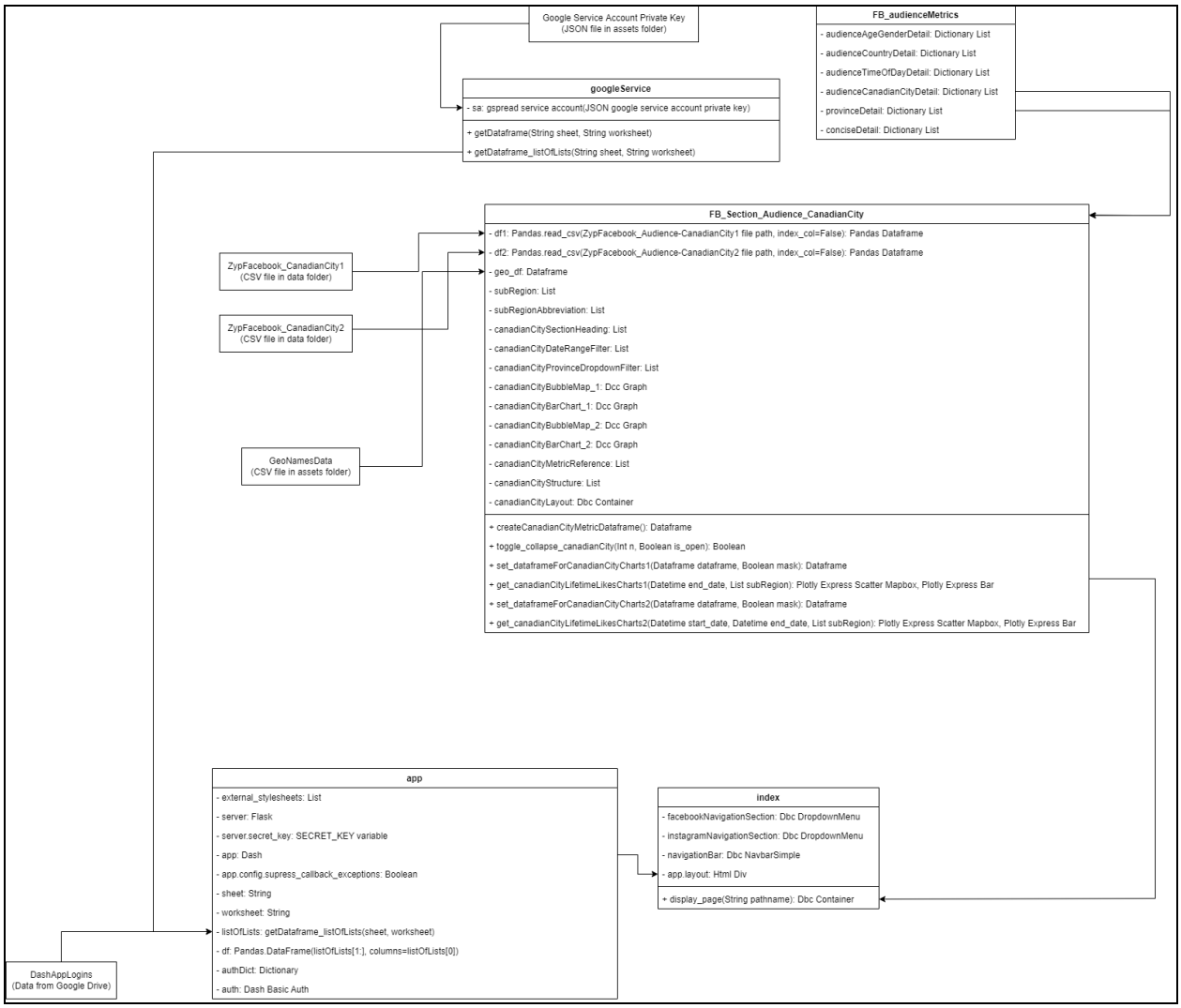
Facebook Audience Insights - Age & Gender (apps/FB_Section_Audience_AgeGender.py)



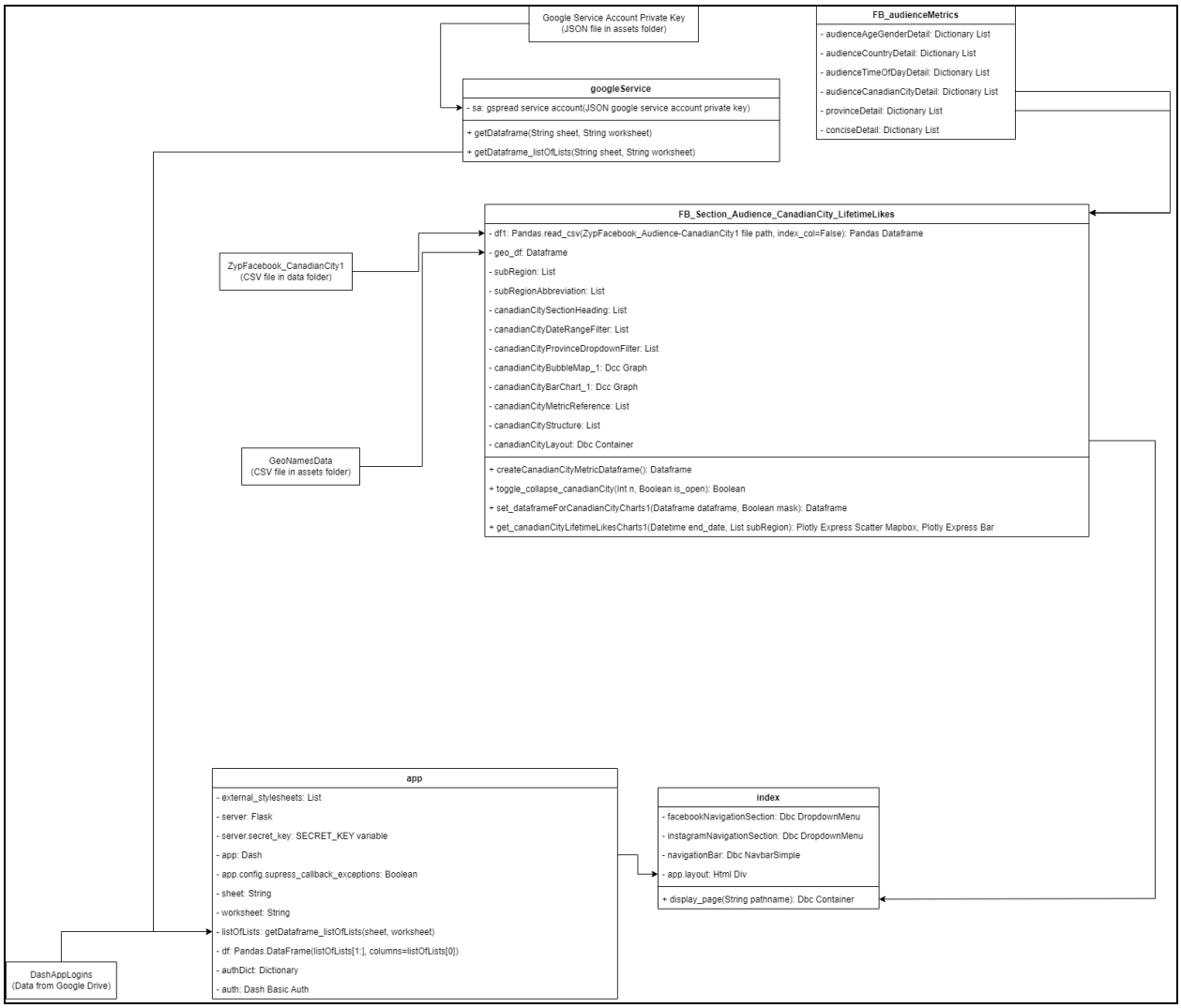
Facebook Audience Insights - Country (apps/FB_Section_Audence_Country)



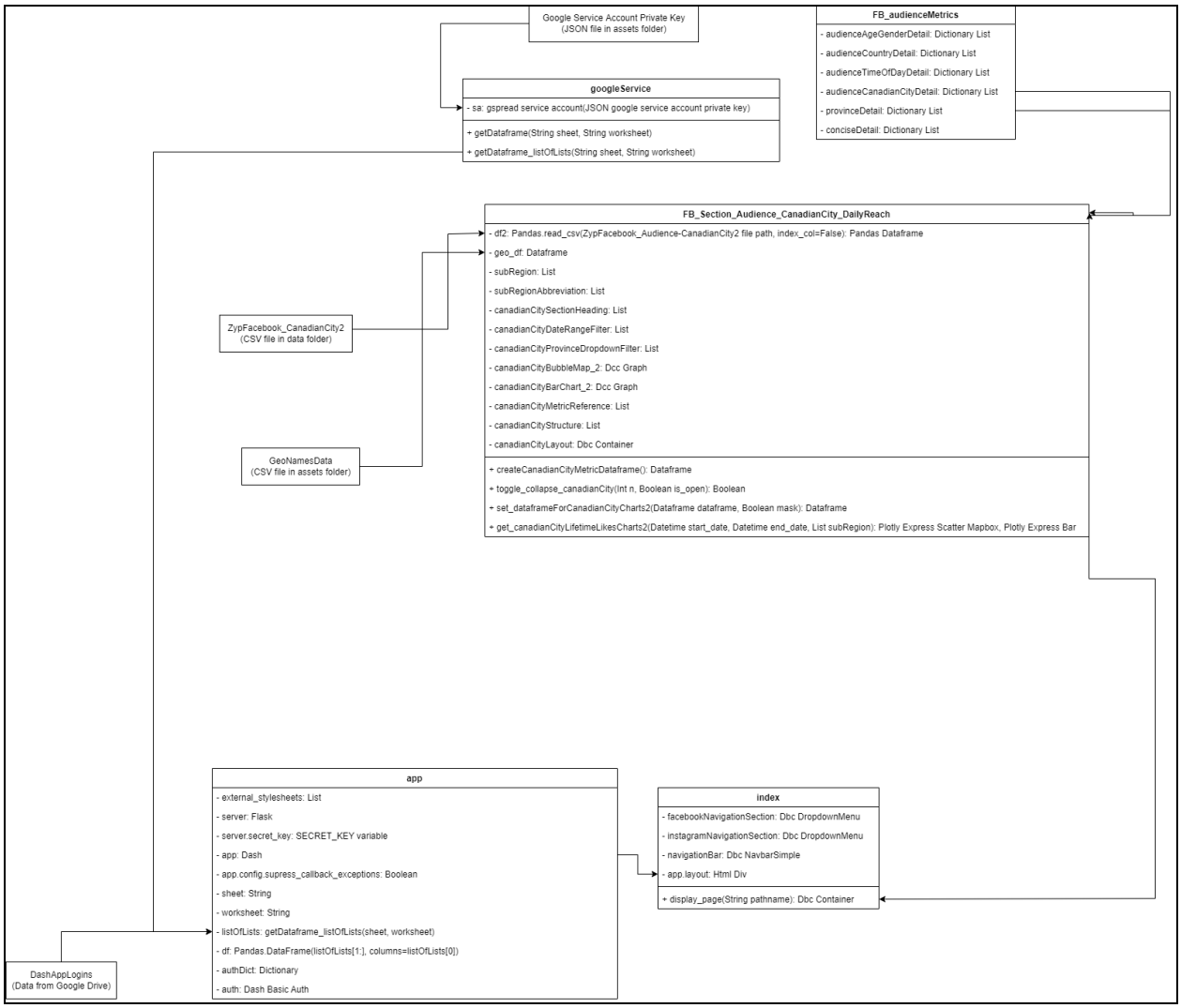
Facebook Audience Insights - Canadian City (apps/FB_Section_Audience_CanadianCity.py)
DEPRECATED



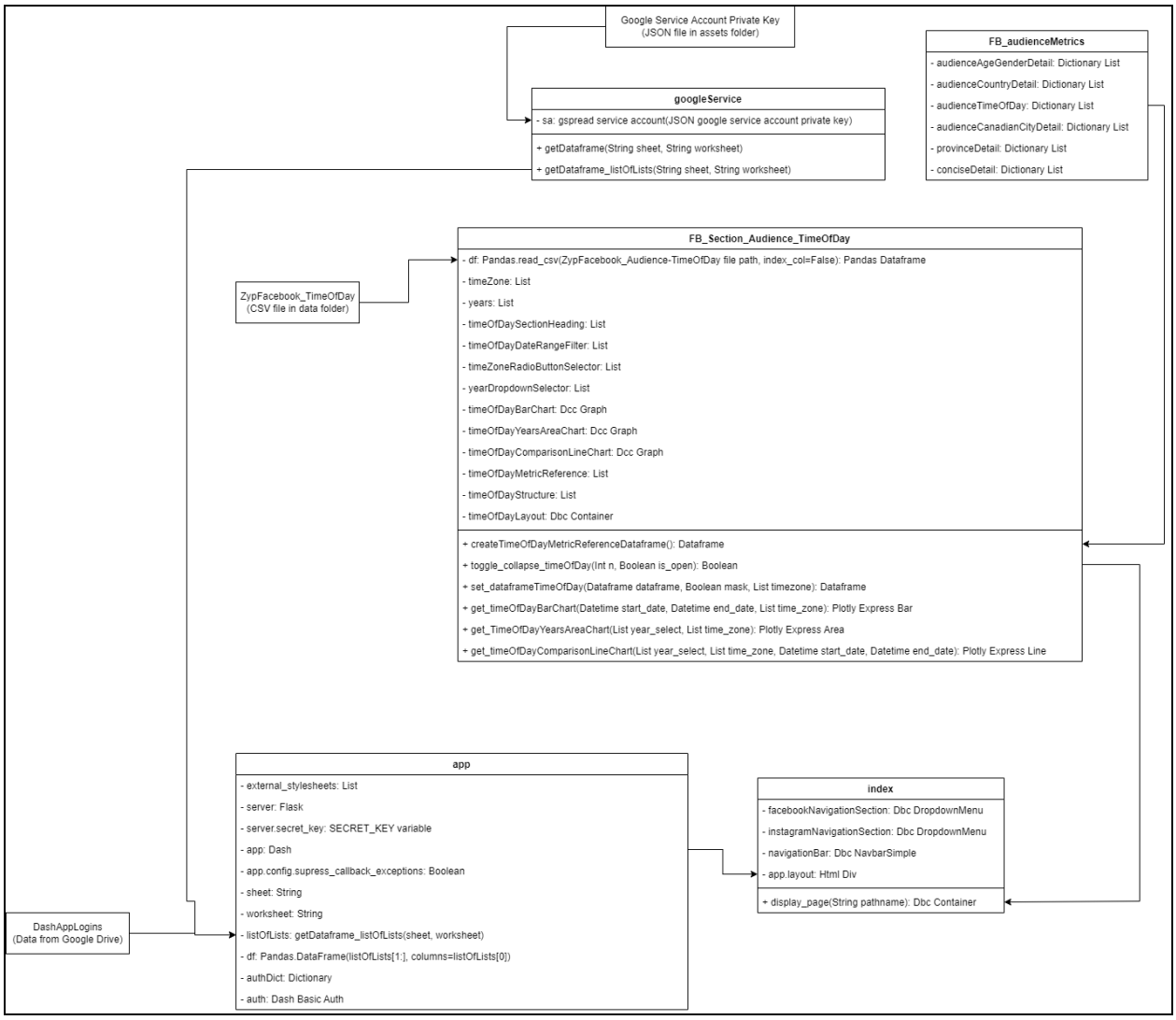
Facebook Audience Insights - Canadian City: Lifetime Likes
(apps/FB_Section_Audience_CanadianCity_LifetimeLikes.py)



Facebook Audience Insights - Canadian City: Daily Reach
(apps/FB_Section_Audience_CanadianCity_DailyReach.py)

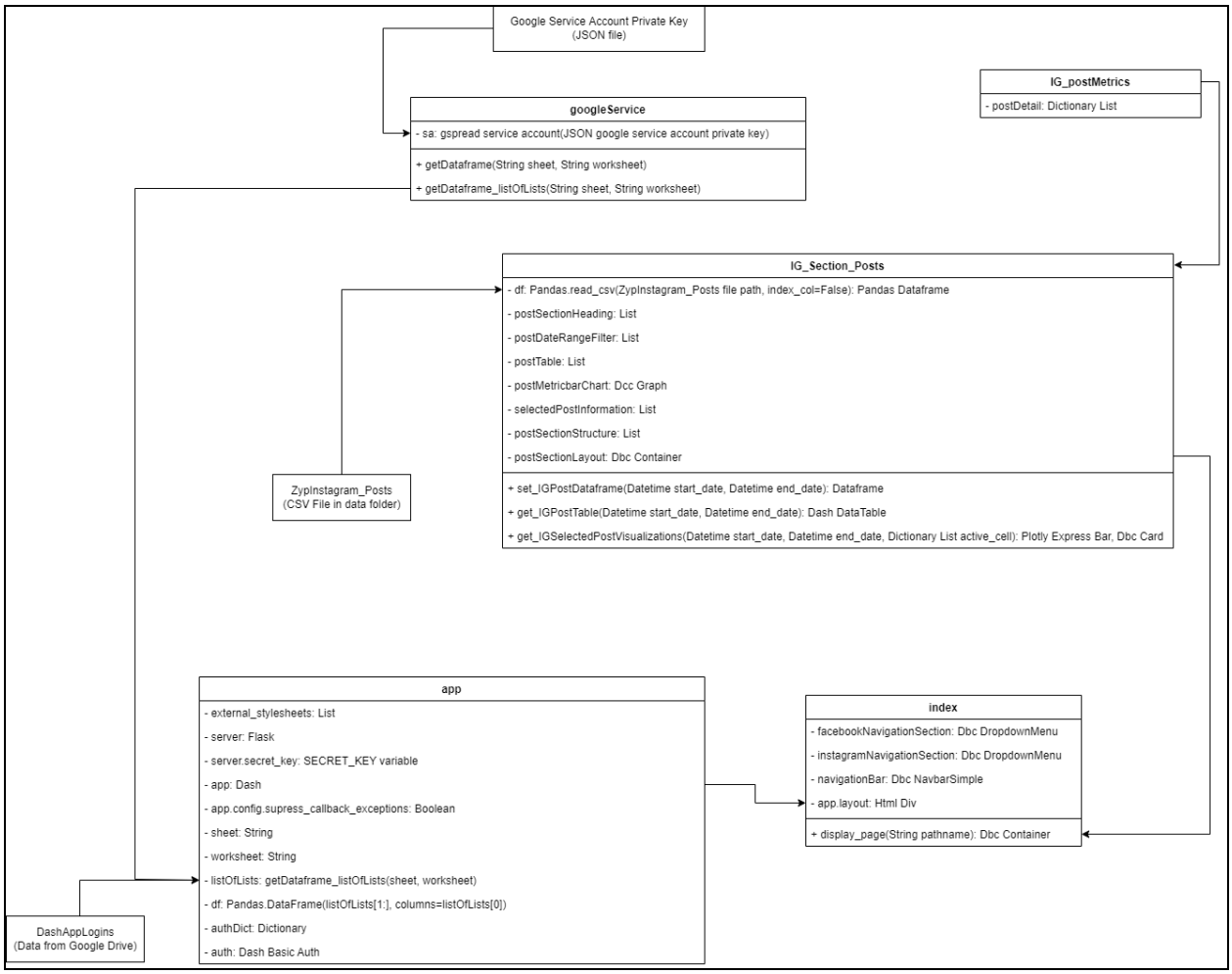


Facebook Audience Insights - Time of Day (apps/FB_Section_Audience_TimeOfDay.py)

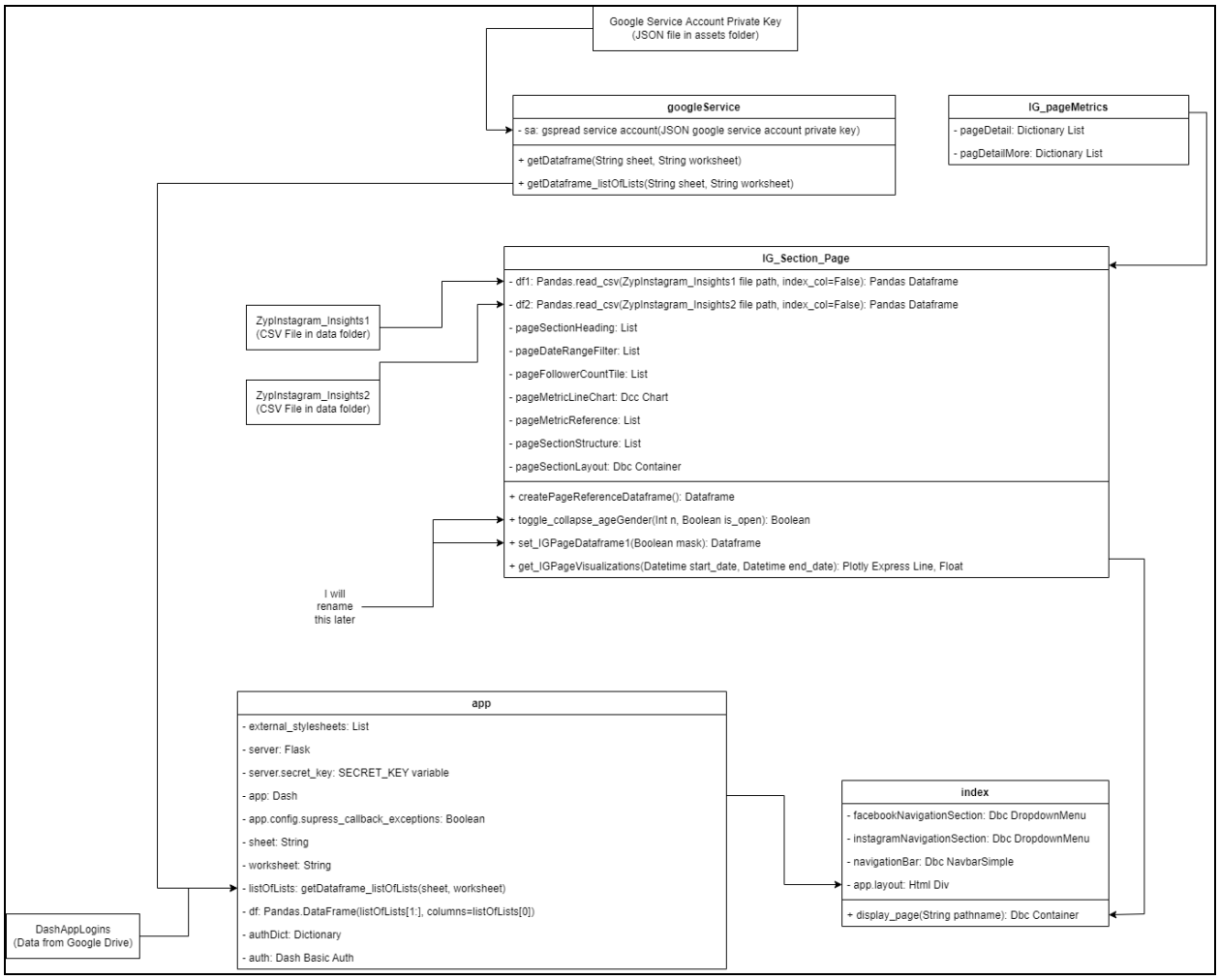


Instagram Sections

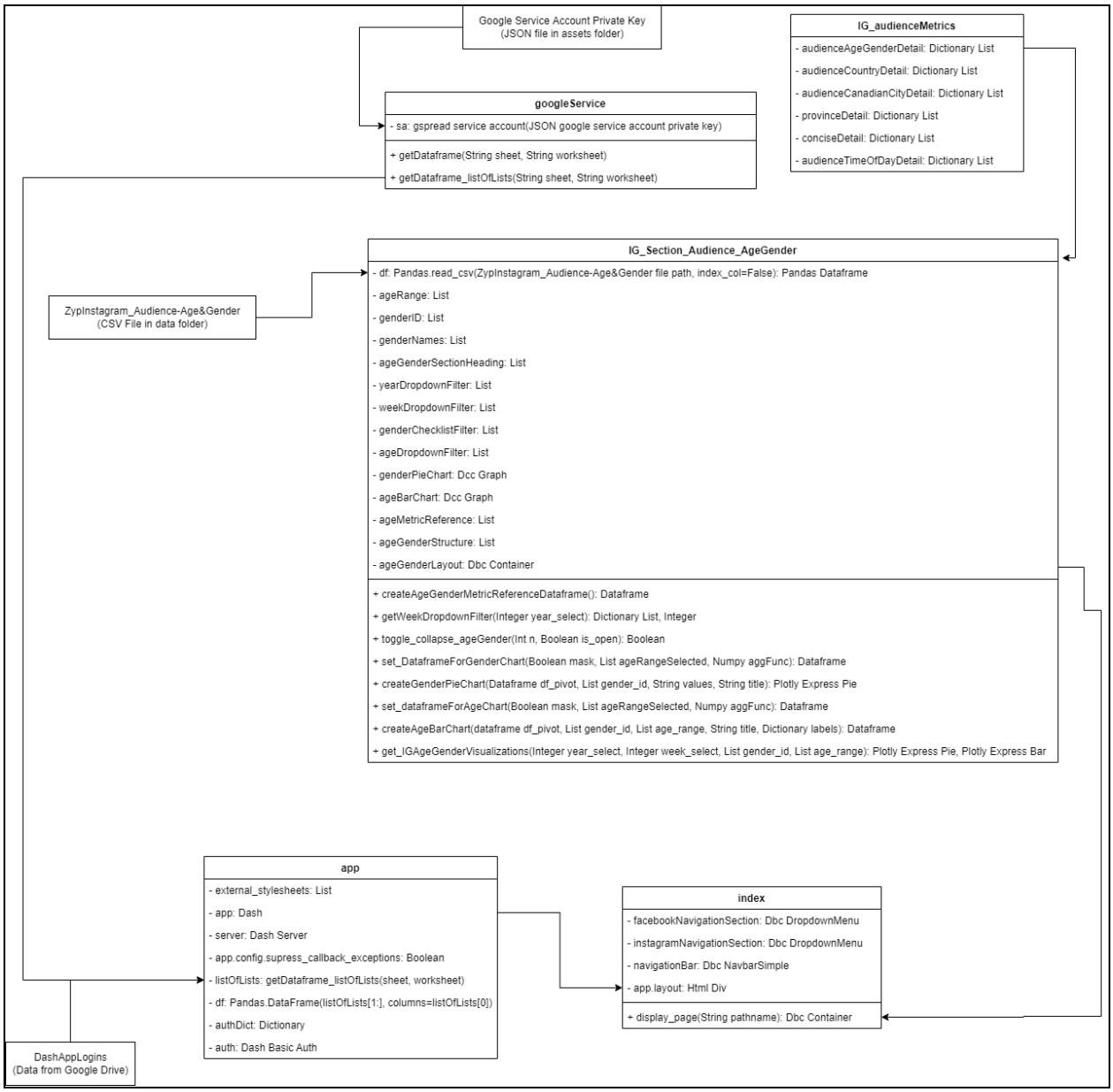
Instagram Post Insights (apps/IG_Section_Posts.py)



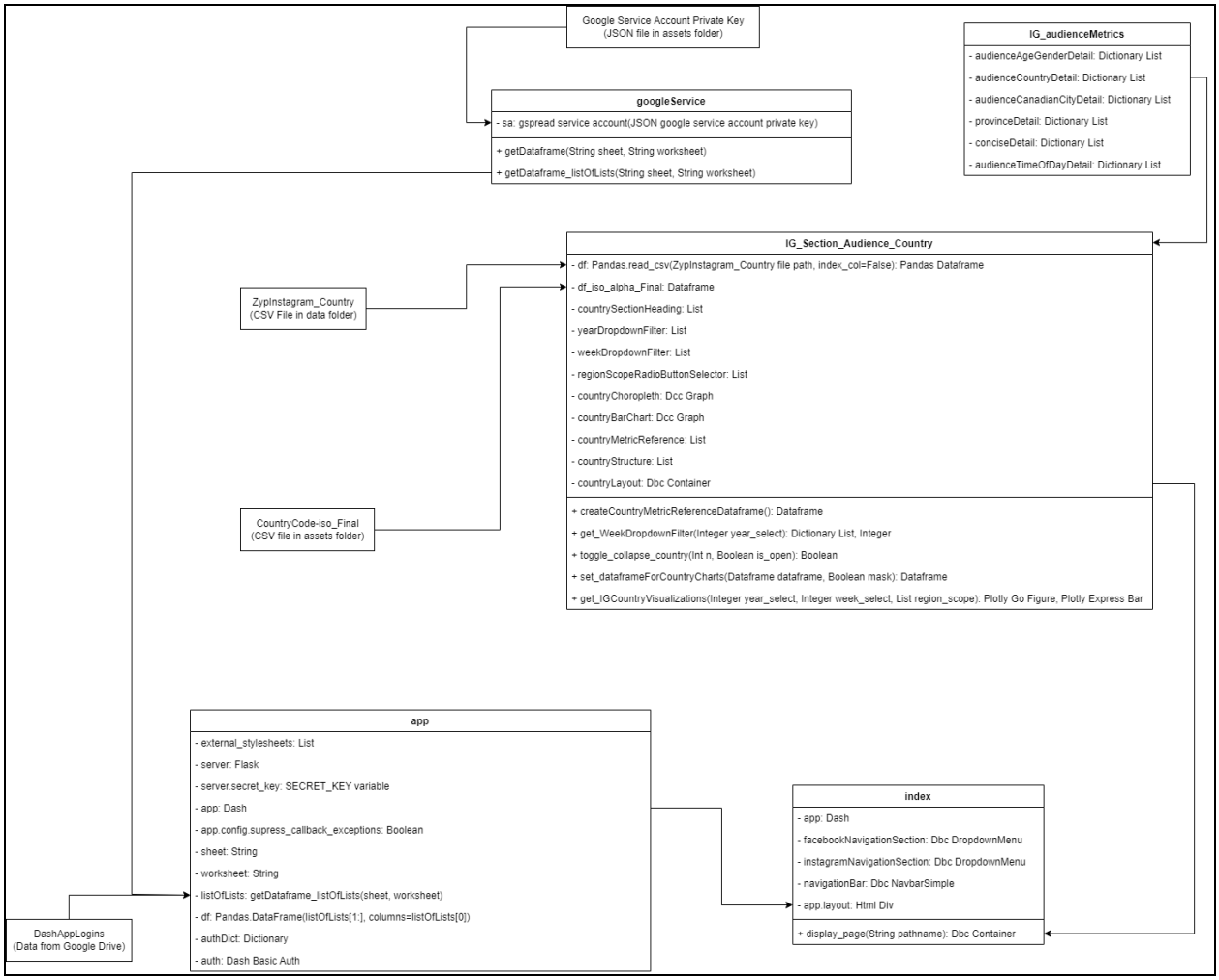
Instagram Page Insights (apps/IG_Section_Page.py)



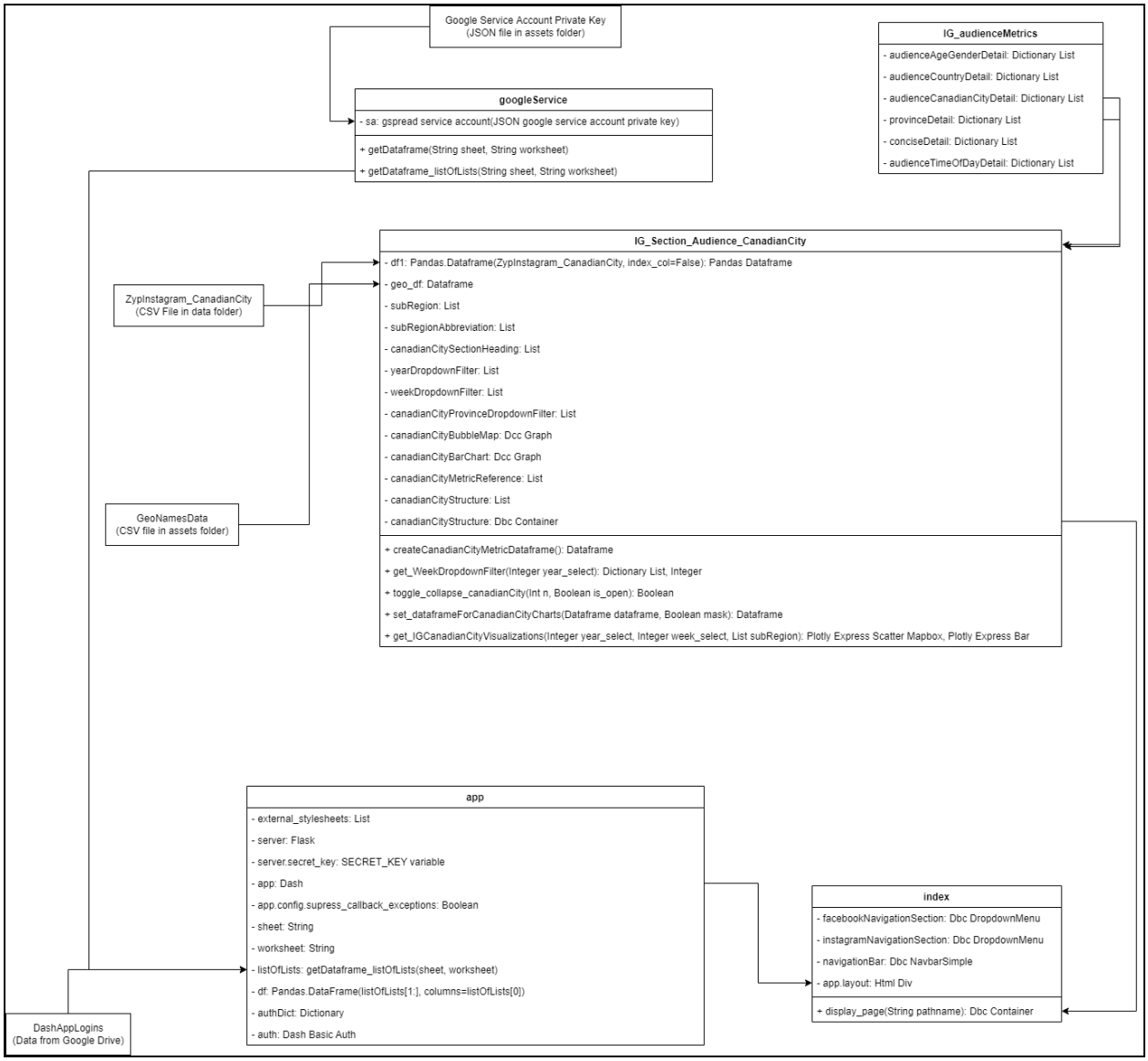
Instagram Audience Insights - Age & Gender (apps/IG_Section_Audience_AgeGender.py)



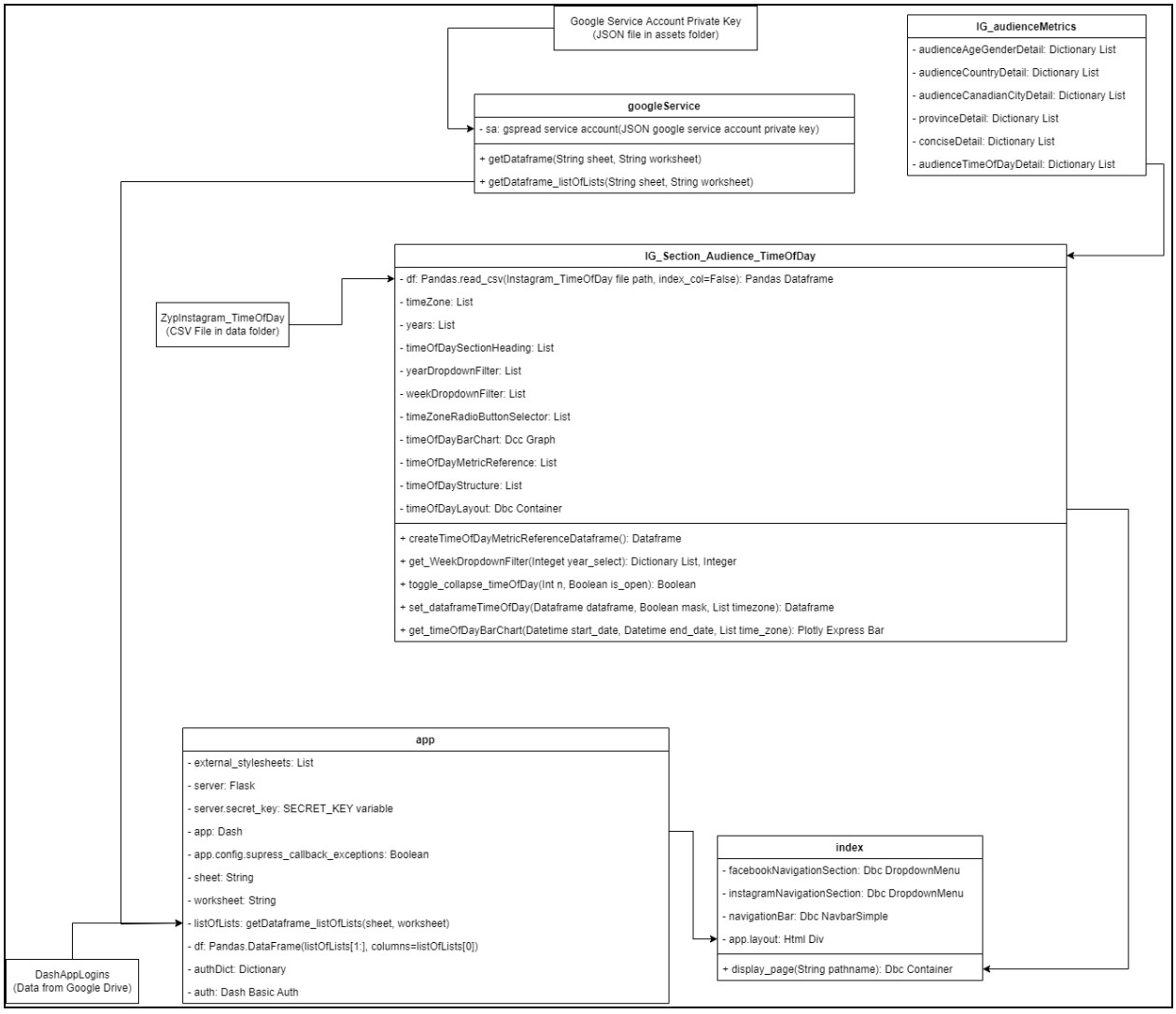
Instagram Audience Insights - Country (apps/IG_Section_Auidence_Country.py)



Instagram Audience Insights - Canadian City (apps/IG_Section_Audience_CanadianCity.py)



Instagram Audience Insights - Time of Day (apps/IG_Section_Audience_TimeOfDay.py)



Design Pattern

As explained in the earlier sections of the report, the main tool of use would be Dash, which is an “open-source library that facilitates the building of interactive web applications” using the Python programming language. The Dash library incorporates the Plotly package which is used to visualize data in a way that is akin to Power BI and Tableau charts & diagrams. The great thing about Dash & Plotly is that you do not require any knowledge of JavaScript; only Python is needed which is easier to use and more user-friendly. Bootstrap, “which is an open-source CSS framework used for web development”, is compatible with Dash elements and is thus used to style the Dash app and make it aesthetically pleasing without the need of building an actual CSS stylesheet from scratch (Wan, 2020).

The scope of this project at this current time does not follow a specific software development design pattern in a traditional sense. However, Dash apps contain three key features that are

imperative to constructing and running a web application successfully. Below are explanations for each of the three features.

1. Layout: This is a feature that tells a Dash app's page how you want it to look like. This can include structural elements such as containers, rows, and columns, along with static web elements such as headings & paragraphs (in other words, lines of text), as well as web elements that are rendered with the intent of interactivity such as radio buttons, checklists, and dropdowns. Furthermore, the aforementioned elements can contain styling elements such as font, colors, and spacing.
2. Components: This is a feature that makes the app interactive. The layout feature may be in charge of instantiating and placing interactive web elements (e.g., radio buttons) on a webpage, but the component feature turns it interactive. For example, the radio buttons in the Facebook time of day webpage are there with the intent to change the visualizations based on the time zone selected from the radio button options (i.e., Mountain Time or Pacific Time).
3. Callbacks: This is a feature that takes the actions taken on the components (e.g., radio button), and performs a change to the webpage. In other words, it takes the the input of a component, and renders an output based on the specified input. In the example of the Facebook time of day webpage, the action taken on the time of zone radio button reflects the change on the connected visualizations.

Once all of the features are constructed and instantiated, the final step to turn the Dash app script to a running web application is to instantiate the app itself and run it on a local server on the computer. It is run on a local server for now as at this point in the project, I am still in the thick of the development stage (Wan, 2020).

As this project is a multi-page app, the dashboard application follows a structural pattern. There are designated scripts that are in charge of rendering the content of pre-defined webpage. For example, there is a specific script created to render the Facebook posts section of the web application, and there is another specific script created to render the Facebook page section of the web application. Each script created to render a section of the web application are in a single folder called "apps". The scripts in the "apps" folder visualize specified CSV files, which contain social media, that are housed in a "data" folder. In addition, there is also an "assets" folder that contains miscellaneous Python scripts that are imperative to the scripts in the "apps" folder, as well as other important files. The contents of the "assets" folder currently consist of the following.

- Google service account private key as a JSON file.
- Google service Python script that takes the Google service account private key and creates a function with the intent to retrieve authentication details (i.e., username & passowrds) from a specified Sheets file from Google Drive.
- Facebook & Instagram metrics Python scripts for post, page, and demographic that contain various dictionary lists of sub-categories containing metric names, titles, and description. These scripts are helpful to create the metric reference table and hover information on visualizations.
- Dataset about Country code ISOs that is used in the data aggregation and preparation code when rendering the Facebook & Instagram country sections of the web application.

ISO codes make the choropleth to actually make the countries visible and show distinction.

- Dataset containing information about Canadian Cities & Towns such as name, province, longitude, and latitude. This data was downloaded and filtered from the GeoName API accessible on the CIC website. This data helps to perform necessary transformation as well as utilizes longitude & latitudes to render bubble map charts in the Facebook & Instagram canadian city sections.

As explained, the scripts in the “apps” folder render the actual webpages in the web application. All of these scripts follow a similar sequential structure as shown below.

1. Import dependencies
    a. Import Python packages
    b. Import any necessary variables from other Python scripts in the assets folder
2. Import the data
    a. Import the required social media data file/s from the ‘data’ folder as a pandas dataframe
3. Prepare the data
    a. Convert the date related column of imported data from string type to datetime type if necessary
    b. Segment columns of imported data into categories if necessary to accommodate for potential-future filtration
    c. Create option labels to be used for interactive web elements that are related to metric category filtration (if needed)
4. Create elements of the webpage
    a. Variables used to instantiate both static and interactive web elements
    b. Static web elements have fully defined parameters
    c. Interactive web elements such as filtration components and/or visualizations have full or partial parameters along with identification elements used that are used by the callbacks to return a specified output
    d. Most of the web elements instantiated are within a Python list so as to group together related elements together to make it easier to place in the variable that instantiates the structure of the page
5. Page layout
    a. Create the skeleton/structure of the webpage that also puts in the variables that instantiate the elements of the webpage
    b. Instantiate the webpage structure to a variable dedicated to layout
6. Callbacks
    a. Contain various functions designed to take the input of various components and return a specified output

Finally, there are two important Python scripts in the main directory of the project. A script called “index.py” is dedicated to running the entire web application on a local server. Thus, it imports all of the layout related variables from the scripts in the “apps” folder. This script also renders a navigation bar. Each link in the navigation bar will correspond to a script in the “apps” folder. Upon selection of a specific link, the corresponding script’s layout would be rendered in the web application, thus, functioning as a multi-page app. Another script called “app.py” instantiates the

Dash application and server as Flask and Dash variables, respectively. This script also instantiates a Dash Authentication variable that implements basic HTTP authentication with the help of the “googleService.py” from the assets folder. The main directory also contains an “_init_.py” file. (Wan, 2020).

Tests Done

I have addressed the objectives of this project and evaluated whether I have achieved them, which is explained in an earlier section of the report. Thus, I decided to create this section in the perspective of my audience. Specifically, my primary target audience is the Analytics team members. I have been showing my progress to the Analytics team members frequently during the team meetings throughout the semester so they are aware of both my efforts and challenges. On July 28th, 2022 at 5:00PM, I did a final demonstration of the project to receive some opinions on whether I fulfilled my objectives or not. Post-submission of this project, I will still be working on this project for the organization. Thus, the project in its current form may be a final version in the context of the CSIS 4495 course, but an ongoing project in the grand scheme of things for the Zyp Art Gallery organization. Below is a table containing three members of the Zyp Art Gallery Analytics team that volunteer regularly and attend meetings every week. Jaime is the Executive Director, Tanmay is the Analytics team leader, and Sandeep is a fellow Analytics team member. The table explains the respective team member’s paraphrased thoughts on whether I have fulfilled my project objectives.

	Jaime	Tanmay	Sandeep
Is the application easy to access by those that are part of the Zyp Art Gallery organization?	Yes it is	Yes it is. Good job! August 4th 2022 update: Facebook Canadian City section not rendering Lifetime Likes visualizations	Yes it is.
Is the application easy to use even for a non-technical person?	Yes. It is easy to use.	Yes. It looks easy to use.	Yes. It looks easy to use for an Analytics team member. Try to ask a parent or younger person to see if they understand the application to get a more definitive answer regarding the question the context of a non-technical person.

Is the application overall aesthetically pleasing?	Application is pleasing to the eyes.	Application is pleasing to the eyes.	Application is pleasing to the eyes.
Does the application follow a logical design in regards to navigation?	Yes it is good.	Yes it is good. The arrows on the Facebook & Instagram dropdown menus could be larger and more visible.	Yes it is all good.
Are the visualizations detailed & informative, yet easy on the eyes?	Yes all good.	All good. The Instagram Posts table should have a title. Axis labels could be in bold so the user understands the visualized data better.	Yes all good.
Do the visualizations have a good degree of interactivity in terms of date range, category, and/or any other identifier?	Interactivity is good.	Interactivity is good. The Canadian City section runs slower. Try to reduce the dataset size. The row values of older years are not necessary so they can be removed from the data file in regards to the project.	Interactivity is good.

Application Setup on Another Machine

Running the Application on a Local Server

In this section, I will be explaining application setup on another machine in the context of the Zyp Art Gallery Analytics team as they are my project's primary target audience. This is so as the intent of this project is to benefit a non-profit organization internally. Thus the application is not meant for public use, and is not freely accessible.

Step 1: Install Python

The first step is to install Python. It was the sole scripting language used to create the application. Python can be installed from the Python organization's website. The conda distribution of Python can be installed as well from the Anaconda website, although this distribution is more suited for Data Science and may work slower when developing & running

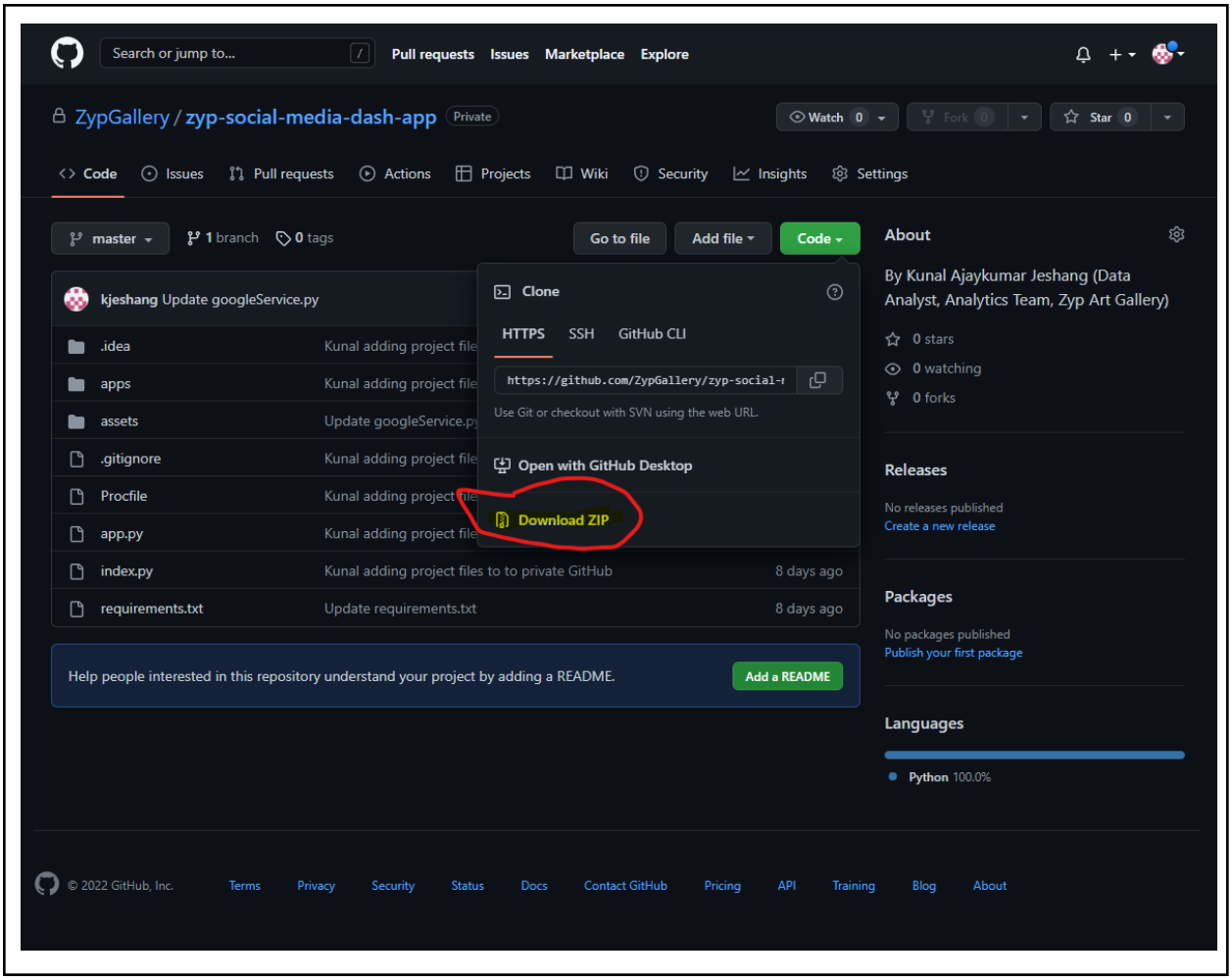
applications. I installed the distribution of Python from its very own website. The version on my Windows computer is 3.10.1, although any version above 3.8.9 should work fine. Below is the download link I would use now if I were to install Python again.
https://www.python.org/downloads/release/python-3105/

Step 2: Pull Project Repository from Zyp Art Gallery's Organization GitHub

The repository is saved on the organization's private GitHub, thus one's GitHub profile must be added to Zyp Art Gallery's organization on GitHub. One would have to be added to the organization on GitHub. The Executive Driector has the ability to do this, and so their permission would be required. They can be contacted via Discord or Zyp Art Gallery email. After one is added to the organization, then they would be able to pull the dashboard application repository into the local machine (i.e., computer).

There are many ways to do this such as using Git commands. Although, the easiest way is to pull a repository to the local machine would be to simply download it. This is a good method for those that simply want to pull repositories just once for use, and it is easily understandable for those that are not well-versed with GitHub or do not use it regularly. After downloading it, the user can save the application's project directory wherever they wish on their local machine. Below is the link to the private repository, as well as some screenshots on how to do this.
https://github.com/ZypGallery/zyp-social-media-dash-app

Below is a screenshot on how one can download a project from GitHub repository directly from the GitHub website.



As the project repository is downloaded as a zipped folder, you have to extract it. In the case of a computer running Windows 11 operating system, follow the instructions under the “To unzip (extract) files or folders from a zipped folder” in the following link. https://support.microsoft.com/en-us/windows/zip-and-unzip-files-8d28fa72-f2f9-712f-67df-f80cf89fd4e5#:~:text=To%20unzip%20a%20single%20file,and%20then%20follow%20the%20instructions.

Step 3: Install a Python IDE, or a code editor with multiple language support

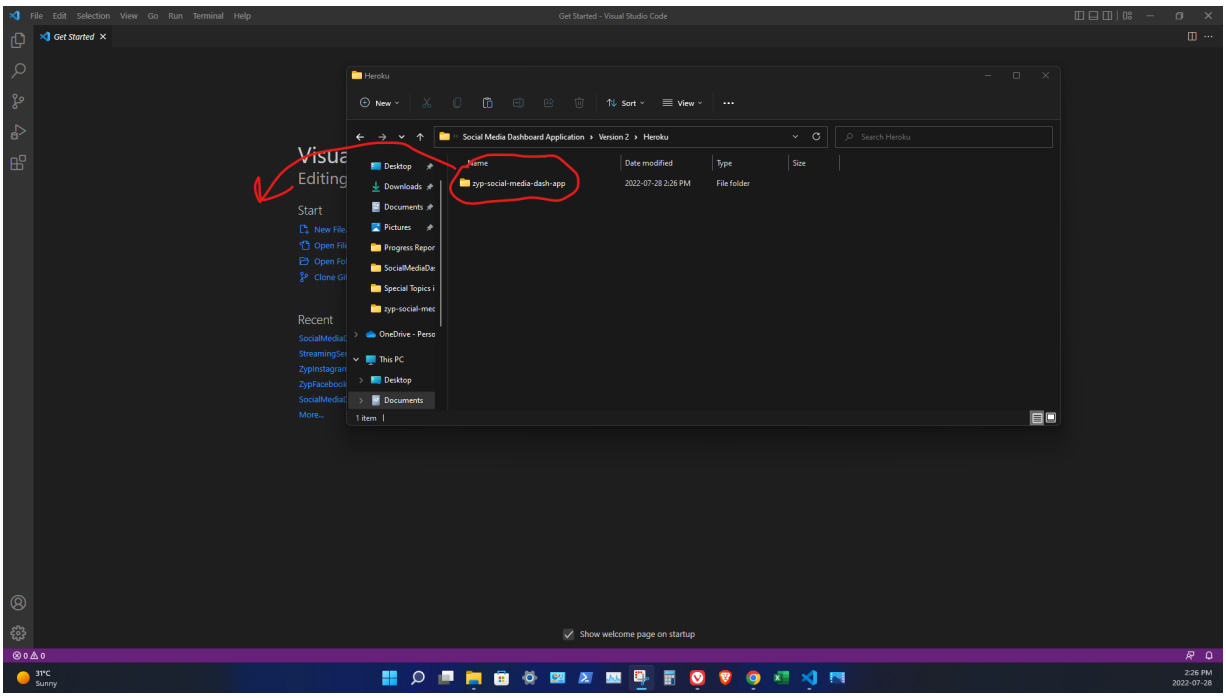
Now that Python is installed. The next step would be to install an IDE or code editor that would be able to open and run the application. The most popular choices in my opinion are Microsoft Visual Studio Code (VS Code) and JetBrains PyCharm Community Edition. I prefer using Microsoft VS Code due to it being a lightweight code editor with compatibility and customizability for various scripting languages. PyCharm is a superior IDE to VS Code, but is very resource intensive. Once installing Microsoft VS Code, the related-required Python-centric add-ons would automatically install upon opening the application’s project directory. Please note the following steps would be using VS Code. Below are download links for VS Code & PyCharm.

- https://code.visualstudio.com/download
- https://www.jetbrains.com/pycharm/download/#section=windows

- The PyCharm installation was quite involved, but the following link was helpful to install the IDE correctly: https://www.youtube.com/watch?v=2RSDlwWH0XI

Step 4: Open the application's project directory in VS Code

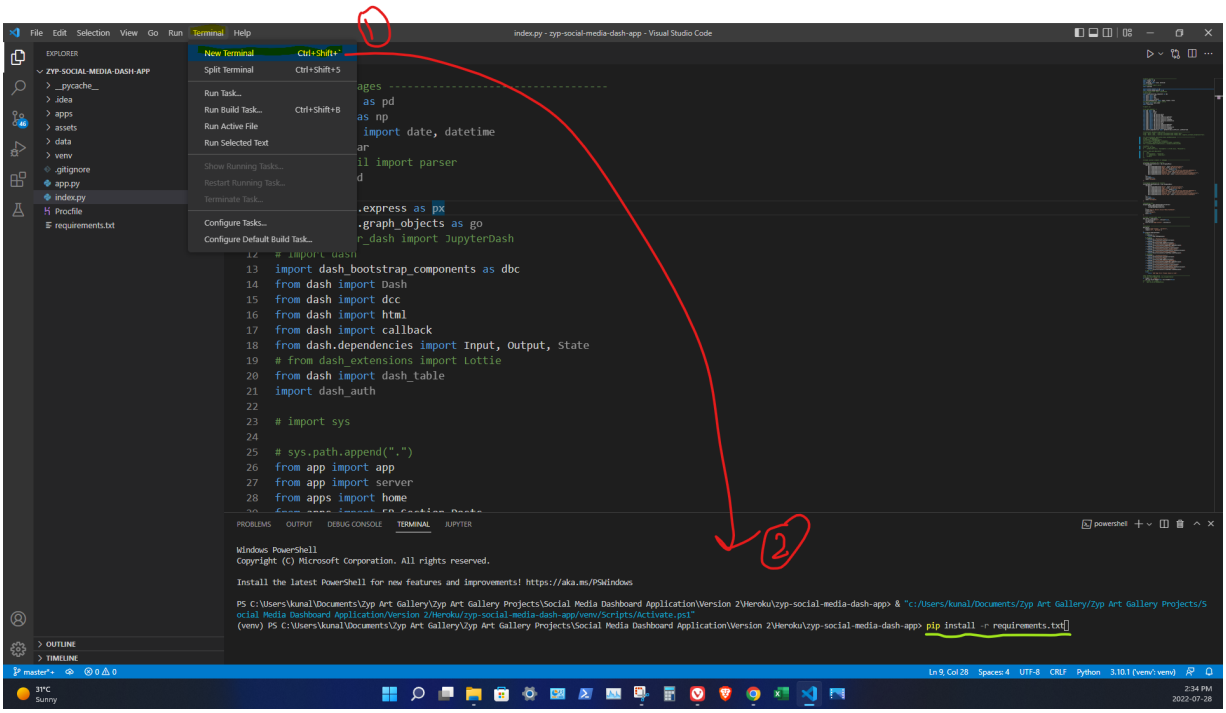
The simplest way to open an application project directory is to first open up VS Code. Then open the directory the project directory is save using the File Explorer. Then simply drag and drop the directory into VS Code. Below is a screenshot for reference. Once the project is opened on VS Code, navigate to the "index" Python script in the root directory.

Below is a screenshot of how you can drag and drop the project folder into VS Code.


Step 5: Install required dependencies

It is imperative to install the required dependencies before attempting to run the application. During the development stage, I created a "requirements.txt" document using 'pip'. This text document contains the names of all the Python packages (i.e., dependencies) used to develop the application as well as the version numbers. Instead of individually installing the each package one at a time, one can simply install all dependencies in one go. Therefore, all one needs to do is create a new terminal. When accessing the terminal whilst a project is open in VS Code, the user is automatically placed in the root directory of the project. Note that I placed the "requirements.txt" file in the root directory of the project. Thus, run the "pip install -r requirements.txt" command within the terminal. After the dependencies have been installed, close the terminal window using by selecting the "trash" icon on the top righ-hand side of the terminal window. Below is a screenshot and Stack Overflow link for reference. https://stackoverflow.com/questions/7225900/how-can-i-install-packages-using-pip-according-to-the-requirements-txt-file-from

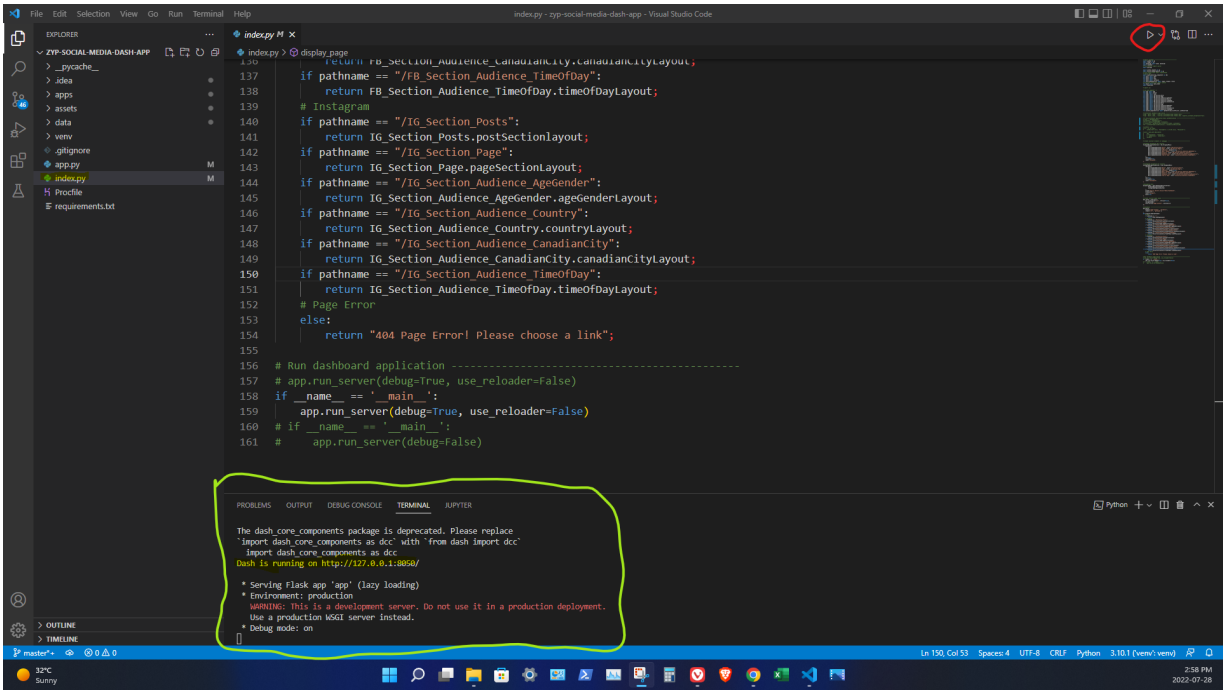
Below is a screenshot that shows the steps that need to be taken to install all required dependencies that are imperative to running the application.



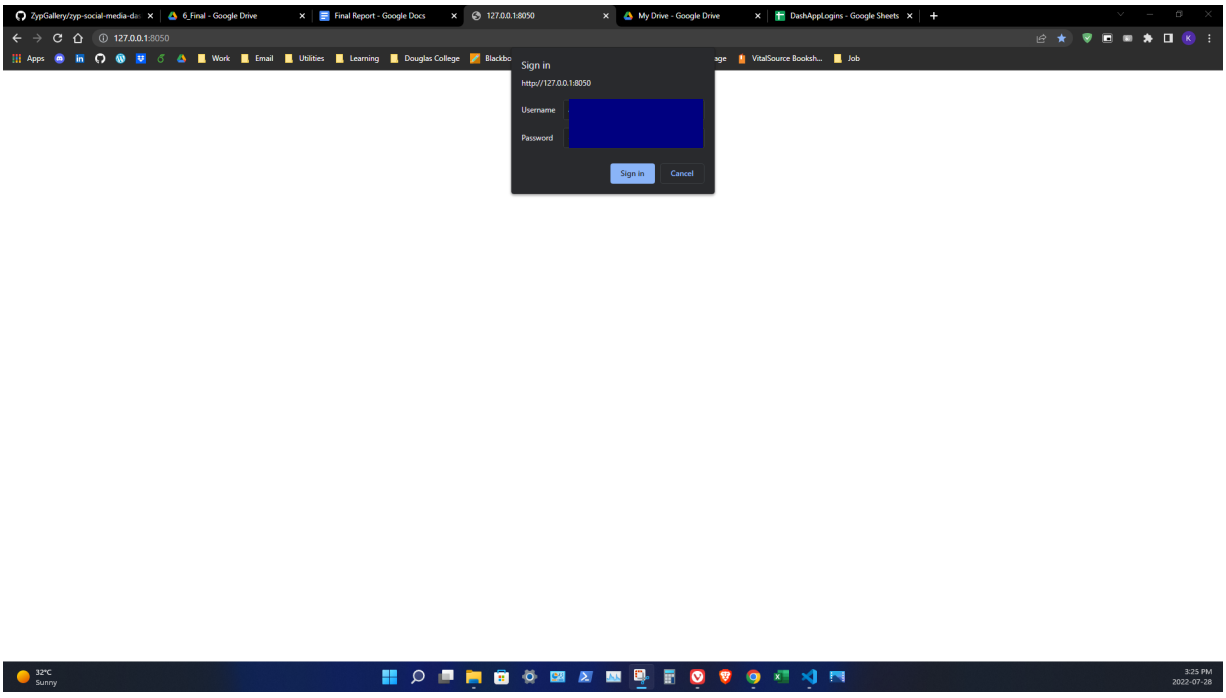
Step 6: Run the application on a local development server

Then simply select the “Play” button on the top right-hand side of the VS Code window. This will run the application on a local server. The terminal will show the link of the local server. One can simply copy paste that link on their web browser, and the application would be accessible. Although, the application itself is requires authentication. Usernames and Passwords are accessible on Zyp Art Gallery’s Google Drive. There is a Google Sheets document that is accessible to only those that are part of the Analytics team containing usernames and passwords. Using the gspread package, the Google Sheets document is imported as a Pandas dataframe and then converted into a dictionary, which is then passed into a Dash Authentication function. Although, in the case of the user, they simply must make sure to put in the correct login credentials. After successful authentication, one would be able to freely use the application within the web browser. If they do not know what the login credentials are, they would have to contact me, the developer of the application, via Discord or Zyp Art Gallery email. If one wants to stop running the application, they simply have to go back to VS Code and simply select the “trash” icon on the top right-hand side of the terminal window. There are some screenshots below to help explain what was just described.

The screenshot below shows how to run the application from VS Code



The screenshot below shows a user would have to authenticate themselves before they can start to use the application.



Below is a screenshot of the home page of the application that appears after successful authentication.

Supplementary Step: Social media data extraction

There is another repository that is available on Zyp Art Gallery's organization GitHub. This repository contains to project folders that contain a various scripts dedicated to extracting data from Facebook & Instagram. This extraction code is run a weekly basis, and the data is appended to the respective Google Sheets documents on the Zyp Art Gallery Google Drive. The data on Google Drive is then imported into the dashboard application for visualization. The extraction code is typically my responsibility to run but any other user could run this as well so that the visualizations are up-to-date when running the dashboard application. One simply needs to download it in the same way as Step 2. Then open 'ZypFacebook' and/or 'ZypInstagram' folders in VS Code in the same way as Step 4. If one wants to extract the Facebook data, then they would run the "FacebookZyp_RunAll" script in VS Code within the 'ZypFacebook' directory; the same way as Step 6. If one wants to extract the Facebook data, then they would run the "InstagramZyp_RunAll" script in VS Code within the 'ZypInstagram' directory; the same way as Step 6. Below is the link to the private repository.
https://github.com/ZypGallery/analytics

Running the Application Online via Heroku

There are not any specific application setup steps when running the application online via Heroku. This is because my primary target audience are simply accessing an already deployed-running version of the application online via Heroku. Thus, the only technology requirement is for the user to have a computer with a working internet connection, and a web browser. They must also know the Heroku link of the application, as well as the login credentials. A prospective user from Zyp Art Gallery would have to contact me via Discord or

email at kunal@zypgallery.ca to find out what the link is as well as the login-authentication details. For simplicity, below is the link to the application. Please note that the submission of this final report will contain login credentials for the instructor.
https://zyp-social-media-dash-app.herokuapp.com/

References

Calmar Art Society — Zyp Art Gallery. (2022). Retrieved 13 May 2022, from https://zypgallery.ca/calmar-art-society

Design Patterns and Refactoring. Sourcemaking.com. (2022). Retrieved 24 June 2022, from https://sourcemaking.com/design_patterns.

Hank and Tillie Zyp — Zyp Art Gallery. (2022). Retrieved 13 May 2022, from https://zypgallery.ca/the-zyps

Honouring Hank and Tillie Zyp — Zyp Art Gallery. (2022). Retrieved 13 May 2022, from https://zypgallery.ca/in-honour

Package Diagram Tutorial. Lucidchart. Retrieved 26 June 2022, from https://www.lucidchart.com/pages/uml-package-diagram#:~:text=a%20UML%20Diagram-,What%20is%20a%20package%20diagram%3F,classes%2C%20or%20even%20other%20packages.

UML Diagram Types Guide: Learn About All Types of UML Diagrams with Examples. Creately. (2021). Retrieved 26 June 2022, from https://creately.com/blog/diagrams/uml-diagram-types-examples/.

Wan, M. (2020). Beginner's Guide to Building a Multi-Page Dashboard using Dash. Medium. Retrieved 24 June 2022, from https://towardsdatascience.com/beginners-guide-to-building-a-multi-page-dashboard-using-dash-5d06dbfc7599.

what we stand for — Zyp Art Gallery. (2022). Retrieved 13 May 2022, from https://zypgallery.ca/values

Zyp art gallery. Zyp Art Gallery. (n.d.). Retrieved May 12, 2022, from https://zypgallery.ca/

Helpful Resources

- Python programming with Plotly Dash package
    - https://realpython.com/python-dash/
    - https://dash.gallery/Portal/
    - https://www.the-analytics.club/plotly-dashboards-in-python

    - https://plotly.com/python/
    - http://dash-bootstrap-components.opensource.faculty.ai/docs/components/
    - https://dash.plotly.com/urls
    - https://www.youtube.com/channel/UCqBFsuAz41sqWcFjZkqmJqQ
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/blob/master/Analytic_Web_Apps/Linkedin_Analysis/jup-d-final-analysis.ipynb
- Country codes for choropleth
    - https://github.com/lukes/ISO-3166-Countries-with-Regional-Codes
    - https://raw.githubusercontent.com/plotly/datasets/master/2014_world_gdp_with_codes.csv
- Social media analytics
    - https://developers.facebook.com/docs/graph-api/reference/v11.0/insights
    - https://developers.facebook.com/docs/instagram-api/reference/ig-media/insights/
- Python programming with GSpread package
    - https://github.com/burnash/gspread
    - https://readthedocs.org/projects/gspread-pandas/downloads/pdf/latest/
    - https://docs.gspread.org/en/latest/user-guide.html#getting-all-values-from-a-worksheet-as-a-list-of-listshttps://github.com/OCA/partner-contact/issues/826
    - https://github.com/burnash/gspread/issues/1007
- Dash bootstrap layout
    - https://dash-bootstrap-components.opensource.faculty.ai/docs/components/layout/
    - https://dash-bootstrap-components.opensource.faculty.ai/docs/components/
- Dash bootstrap Card & CardBody
    - https://dash-bootstrap-components.opensource.faculty.ai/docs/components/card/
    - https://www.youtube.com/watch?v=aEz1-72PKwc
    - https://developer.mozilla.org/en-US/docs/Web/CSS/font-size
- Dash bootstrap navigation bar and tabs
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/blob/master/Bootstrap/Side-Bar/side_bar.py
    - http://dash-bootstrap-components.opensource.faculty.ai/docs/components/navbar/
    - https://community.plotly.com/t/adding-callback-to-dashmenuitem-of-dash-navbar-component/38815
    - https://dash-bootstrap-components.opensource.faculty.ai/docs/components/tabs/
    - https://dash.plotly.com/dash-core-components/tabs
    - https://dash-bootstrap-components.opensource.faculty.ai/docs/components/dropdown_menu/
- Dash Datatable
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/blob/master/DataTable/datatable_intro_and_sort.py
    - https://dash.plotly.com/datatable
    - https://github.com/plotly/dash-table
    - https://community.plotly.com/t/how-to-wrap-text-in-cell-in-dash-table/15687/6

    - https://dash.plotly.com/datatable/tooltips
- Multiple pages
    - https://dash.plotly.com/urls
    - http://dash-bootstrap-components.opensource.faculty.ai/examples/simple-sidebar/
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/blob/master/Deploy_App_to_Web/Multipage_App/index.py
- Authentication
    - https://levelup.gitconnected.com/how-to-setup-user-authentication-for-dash-apps-using-python-and-flask-6c2e430cdb51
    - https://www.youtube.com/watch?v=MxQtgLVEqbQ&list=WL&index=4
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/tree/master/Deploy_App_to_Web/Authentication
    - https://dash.plotly.com/authentication
    - https://www.analyticsvidhya.com/blog/2021/05/create-login-page-in-dash-library/
    - https://www.youtube.com/watch?v=j3VvVaNnDH4
- Python file structure
    - https://stackoverflow.com/questions/4383571/importing-files-from-different-folder
    - https://www.kite.com/python/answers/how-to-import-a-class-from-another-file-in-python
- Dash datepicker
    - https://dash.plotly.com/dash-core-components/datepickerrange
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/blob/master/Dash%20Components/DatePickerRange/datepicker.py
- Python programming (date & time, dictionary list)
    - https://www.kite.com/python/answers/how-to-subtract-days-from-a-date-in-python
    - https://www.analyticsvidhya.com/blog/2021/06/working-with-lists-dictionaries-in-python/
    - https://docs.python.org/3/tutorial/datastructures.html
    - https://stackoverflow.com/questions/3476732/how-to-loop-backwards-in-python
- Dash server
    - https://stackoverflow.com/questions/59568510/dash-suppress-callback-exceptions-not-working
- Dash callbacks
    - https://stackoverflow.com/questions/59382089/how-to-use-a-callback-to-add-to-a-string-stored-in-a-div-using-dash-in-python
    - https://dash.plotly.com/basic-callbacks
- Plotly elements
    - https://plotly.com/python/hover-text-and-formatting/
    - https://plotly.com/python/figure-labels/
    - https://stackoverflow.com/questions/58166002/how-to-add-caption-subtitle-using-plotly-method-in-python
    - https://stackoverflow.com/questions/46512682/how-to-set-the-bold-font-style-in-plotly

- Pandas
    - https://www.codegrepper.com/code-examples/python/replace+empty+string+with+0+pandas
    - https://sparkbyexamples.com/pandas/pandas-remove-duplicate-columns-from-dataframe/
    - https://stackoverflow.com/questions/32444138/concatenate-a-list-of-pandas-dataframes-together
    - https://www.geeksforgeeks.org/python-pandas-timestamp-year/
    - https://stackoverflow.com/questions/16923281/writing-a-pandas-dataframe-to-csv-file
    - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html
    - https://stackoverflow.com/questions/37954195/upgrade-version-of-pandas
    - https://stackoverflow.com/questions/54898630/weird-error-when-dividing-two-numbers-in-pandas-dataframe
    - https://datatofish.com/rows-with-nan-pandas-dataframe/
- Dashboard design on Dash
    - https://towardsdatascience.com/create-a-professional-dasbhoard-with-dash-and-css-bootstrap-e1829e238fc5
    - https://towardsdatascience.com/beginners-guide-to-building-a-multi-page-dashboard-using-dash-5d06dbfc7599
    - https://medium.com/swlh/dashboards-in-python-3-advanced-examples-for-dash-beginners-and-everyone-else-b1daf4e2ec0a
- Dash dropdown
    - https://dash.plotly.com/dash-core-components/dropdown
    - https://stackoverflow.com/questions/62859122/dash-callback-with-dropdown
- Dash bootstrap tooltip
    - http://dash-bootstrap-components.opensource.faculty.ai/docs/components/tooltip/
- Dash bootstrap collapse
    - http://dash-bootstrap-components.opensource.faculty.ai/docs/components/collapse/
- Table
    - http://dash-bootstrap-components.opensource.faculty.ai/docs/components/table/
- Plotly choropleth chart
    - https://towardsdatascience.com/geographical-plotting-of-maps-with-plotly-4b5a5c95f02a
    - https://plotly.com/python/map-configuration/
    - https://plotly.com/python/bubble-maps/
    - https://www.iso.org/obp/ui/#iso:code:3166:AN
- Plotly bar chart
    - https://plotly.com/python/horizontal-bar-charts/
- Dash radiobutton
    - https://dash.plotly.com/dash-core-components/radioitems
- Plotly area chart

    - https://chartio.com/learn/charts/area-chart-complete-guide/#:~:text=You%20will%20use%20a%20stacked,their%20contributions%20to%20the%20total.
    - https://plotly.com/python/filled-area-plots/
- Sources for longitude and latitude codes for Canadian city locations
    - https://www.nrcan.gc.ca/maps-tools-and-publications/maps/geographical-names-canada/application-programming-interface-api/9249
    - https://www.nrcan.gc.ca/earth-sciences/geography/download-geographical-names-data/9245
    - https://www.latlong.net/category/provinces-40-60.html
    - https://www.geodatos.net/en/coordinates/canada
    - https://www.latlong.net/place/nunavut-canada-25380.html
    - https://www.latlong.net/place/yukon-territory-canada-29628.html
    - https://latitude.to/articles-by-country/ca/canada/1086/northwest-territories
- Plotly bubble map chart
    - https://www.youtube.com/watch?v=7R7VMSLwooo
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/blob/master/Dash_Interactive_Graphs/Scatter_mapbox/recycling.py
    - https://docs.mapbox.com/help/glossary/access-token/
    - https://www.youtube.com/watch?v=1-6ndLqsy6M
    - https://plotly.github.io/plotly.py-docs/generated/plotly.express.scatter_mapbox.html
    - https://github.com/plotly/dash/issues/914
    - https://plotly.github.io/plotly.py-docs/generated/plotly.express.scatter_mapbox.html
    - https://community.plotly.com/t/define-title-and-legend-place-on-a-px-scatter-mapbox/42300/2
- Dash bootstrap button
    - https://stackoverflow.com/questions/64106370/adding-buttons-with-external-links-to-plotly-dash-modal
    - http://dash-bootstrap-components.opensource.faculty.ai/docs/components/button/
- Dash deployment to Heroku
    - https://towardsdatascience.com/beginners-guide-to-building-a-multi-page-dashboard-using-dash-5d06dbfc7599
    - https://www.youtube.com/watch?v=Gv910_b5ID0&list=PLh3I780jNsiShH-OYSfosz2x1KzRm_F_U&index=7
    - https://towardsdatascience.com/deploying-your-dash-app-to-heroku-the-magical-guide-39bd6a0c586c
    - https://www.youtube.com/watch?v=j3VvVaNnDH4
    - https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/How-to-push-an-existing-project-to-GitHub
    - https://towardsdatascience.com/deploying-your-dash-app-to-heroku-the-magical-guide-39bd6a0c586c

    - https://stackoverflow.com/questions/24114676/git-error-failed-to-push-some-refs-to-remote
    - https://python.plainenglish.io/how-to-convert-your-dash-app-into-an-executable-gui-b1d4271a8fa7
    - https://help.heroku.com/LGKL6LTN/how-do-i-delete-destroy-a-heroku-application
    - https://dev.to/jovialcore/heroku-solved-failed-to-push-some-refs-486p
    - https://stackoverflow.com/questions/50026190/heroku-fails-to-install-pywin32-library
    - https://www.youtube.com/watch?v=RMBSQ6leonU&list=PLh3I780jNsiShH-OYSfosz2x1KzRm_F_U&index=3
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/tree/master/Deploy_App_to_Web/Multipage_App
    - https://github.com/Coding-with-Adam/Dash-by-Plotly/tree/master/Deploy_App_to_Web/Authentication
    - https://stackoverflow.com/questions/63880206/how-to-run-bootstrap-css-components-in-heroku-python-dash-application
    - https://stackoverflow.com/questions/61452429/plotly-dash-app-gives-error-after-deployment-to-heroku
    - https://community.plotly.com/t/display-favicon-in-heroku-dash-app/44013/4
    - https://favicon.io/favicon-converter/
    - https://github.com/dc-aichara/DS-ML-Public/tree/master/Medium_Files/dashboard_demo
    - https://www.angela1c.com/posts/2021/09/deploying-dash-apps-to-heroku/
- Dash bootstrap Jumbotron
    - https://dash-bootstrap-components.opensource.faculty.ai/docs/components/jumbotron/
    - https://community.plotly.com/t/how-to-embed-images-into-a-dash-app/61839
    - https://community.plotly.com/t/clickable-a-tag-on-image-source-in-dash/10016/2

Appendix

Progress Report 3

Post-submission of progress report 2 until the time of this section's writing (i.e., July 11th to July 25th), I managed to implement a basic authentication method to the dash application, as well as researched and attempted deployment of the application to Heroku. I adjusted functions used by various scripts in the 'app' subdirectory to try to accommodate for Google Sheets API threshold limit. This was a challenging period of working on the project as there is less time between now and the submission date to make large changes. The application functions and is relatively usable.

As explained in the previous project report, I had to apply different gspread functions to help manage Google Sheets API request threshold limit. I utilized a function that retrieves the social

media data as a list of lists rather than retrieval of all records, and then I convert into a pandas dataframe. Essentially, it is the same function used in the Canadian City related sections. This helped to manage the threhold limit to a certain degree.

In regards to authentication, it was my original desire to implement Google Authentication to the application if it were deployed online. As Dash is built on top of Flask, I believed I could find good resources to learn from. My research of finding precise instructions to step-by-step implement this did not bear fruit. The sources that I did refer to lacked clarity and were not precise enough for me. Therefore, I decided to utilize basic HTTP authentication, which is built into Dash, to that renders a built-in browser pop-up asking for login credentials. Instead of hard coding usernames and passwords into a tuple within an instantiated dash authentication function, I created a Google Sheets document saved on the organization's Google Drive that contained usernames and passwords. I read the Google Sheets document as a pandas dataframe, which is in a similar way to how the social media data is being read into the application. The pandas dataframe is then converted into a dictionary, and then incorporated into the dash authentication function.

I also managed to create a home/landing page for the dashboard application. It is very simple that has a welcome message, my name, and briefly explains what the application is used for. This page is the default path of the application, and is hyperlinked to the title of the application which is on the navigation bar. This page also has the logo of the organization. For this part of the application, I did not spend time creating a user interface design. For now, I just wanted to create something simple due to limited time, and also it would take more time to coordinate with the Analytics team & Executive Director.

Finally, I attempted to deploy to the application to Heroku. For that I needed to adjust various components in the app and index scripts. This mainly included changing some settings when running the dash application on a server, as well as importing the server variable instantiated from the app script. More steps were taken such as:

- Installing PyCharm community edition
- Installing GitHub CLI tools
- Installing Heroku CLI tools

I then attempted to recreate the project using PyCharm that incorporated a virtual environment. This is so as the project until this point in time had a structure sufficient for deployment on a local server but not to the web. This way, only the packages that are necessary to the project would be installed when deployed to Heroku. I then used the 'pip freeze > requirements.txt' command using the terminal to get the package names and versions relevant to the project. A key aspect of this to work without retrieving all of my was to make sure that "(venv)" appeared next to the project's root directory. I then had to create ".gitignore" and "Procfile" documents in the root directory of the project. After all of this, I then triggered Heroku & Git commands to attempt to deploy the application to Heroku. Specifically, I performed the following:

1. Logged into Heroku
2. Created a Heroku app that corresponds with the name of the project, which creates a git repo for Heroku

3. Performed git commands to add & commit
4. Pushed the project to the master branch of the git repo
5. Scale the web app to use 1 dyno (free tier)

It should be noted that the name of the project that was pushed to Heroku is different. This is to help me differentiate it from the project that works on a local server. I had to follow written tutorials and video tutorials. A long list of the resources I referred to is in the “Helpful Resources” section of this report. Although, below is a list of the ones I used the most.

- https://towardsdatascience.com/beginners-guide-to-building-a-multi-page-dashboard-using-dash-5d06dbfc7599
- https://www.youtube.com/watch?v=RMBSQ6leonU&list=PLh3I780jNsiShH-OYSfosz2x1KzRm_F_U&index=3
- https://python.plainenglish.io/how-to-convert-your-dash-app-into-an-executable-gui-b1d4271a8fa7
- https://towardsdatascience.com/deploying-your-dash-app-to-heroku-the-magical-guide-39bd6a0c586c

The above links I felt were the best in terms of clarity and information, but unfortunately the application did not deploy successfully to Heroku. Specifically, the application managed to deploy although after opening the application, there were was an application error. As per the instructions displayed on the error page, I used the Heroku command “heroku logs - -tail” to see what the errors were. Due to my inexperience, I was not sure what the precise reason was. The log output said that the application crashed and there was something regarding ‘favicon’ which could have been due to using Dash bootstrap elements package. There could be a number of reasons why my application could not be viewed. According to one of the links above, these are some of the reasons

- Boostrap error due to favicon which is used by the Dash bootstrap elements package
- Timeout of the application does not fully run within 30 seconds
    - Caused by a lot of time elapsing when importing social media data from Google Drive?

This period of development was upsetting for me as even after performing the needful and following the appropriate steps, my application was unable to live online and be truly accessible. At the time of this writing, there are only two more weeks left until final project submission. There may not be much time to further tinker with the project to either successfully live online via Heroku or perform extensive research to make it be accessible through other means. This is because I would require more time to write the final report and prepare any other required deliverables. That being said, the project works on my end when running locally.

As the project is able to be run locally, I decided to push the project to the organizations private GitHub repository so that it can be accessed and viewed by other members & volunteers of the organization.

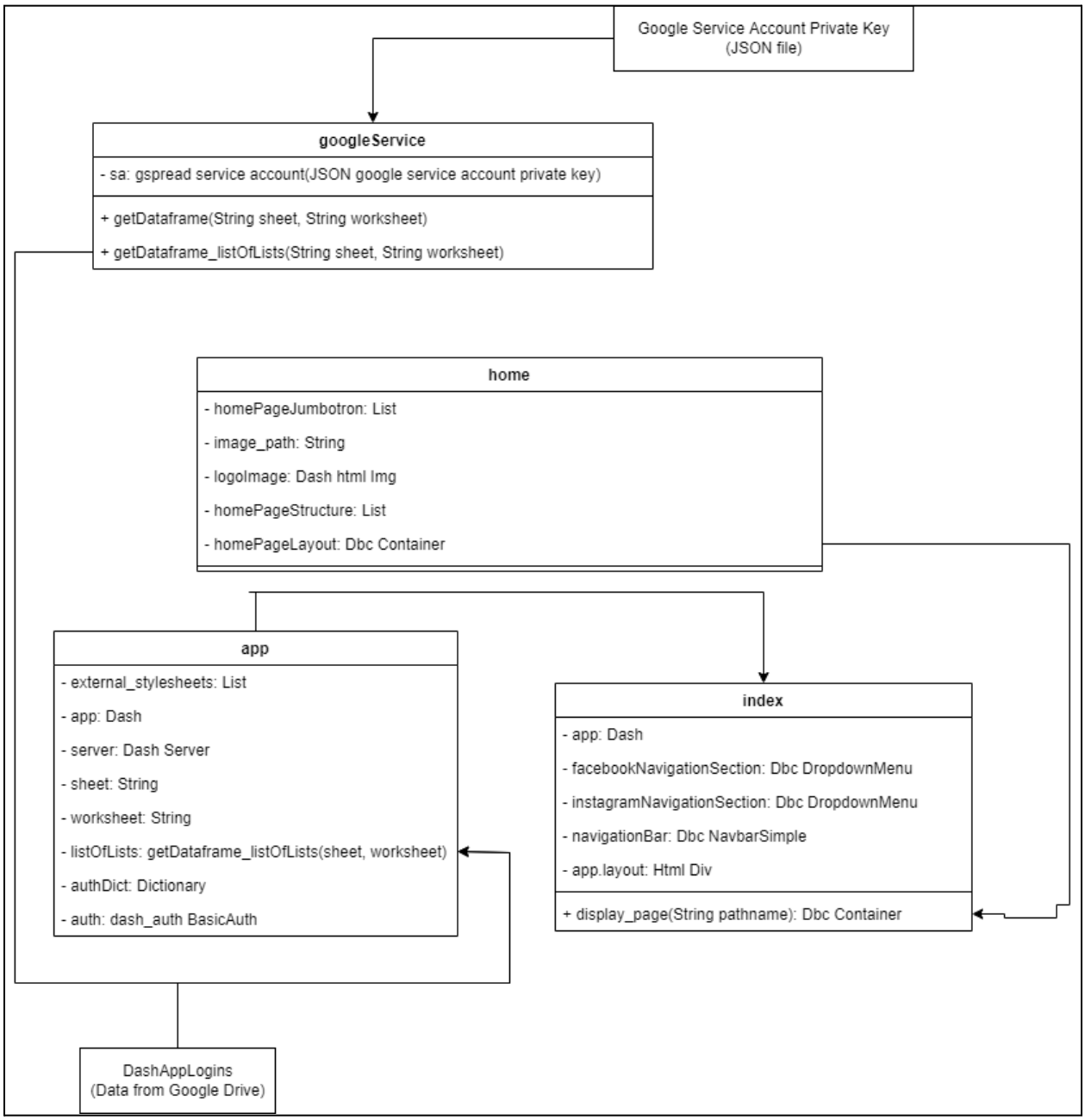
The following section provides evidence on majority of the progress that was described in the prior paragraphs.

Evidence

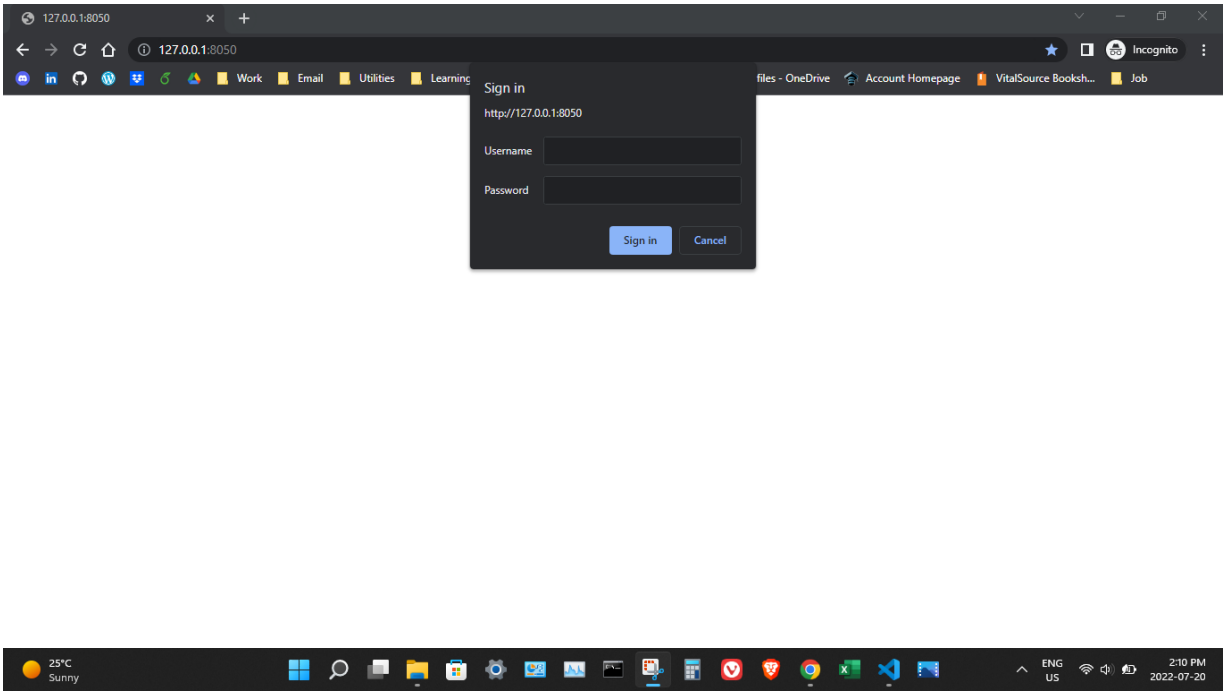
UML

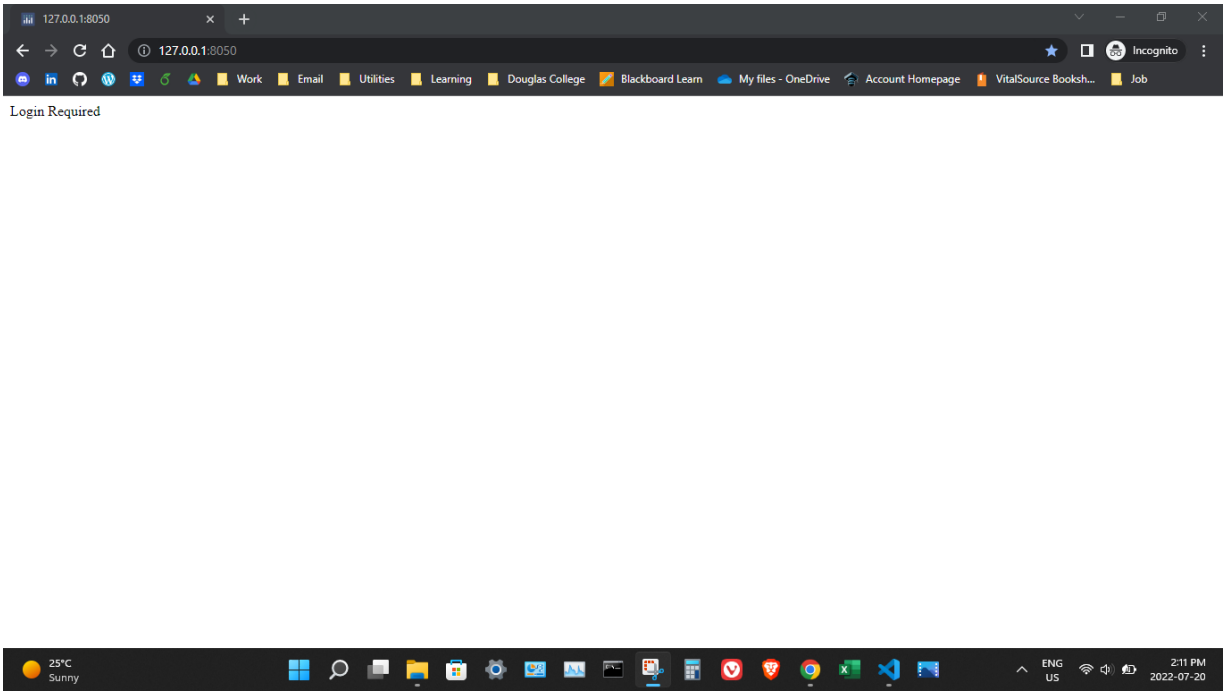
As all of the UML diagrams for the other webpages is complete, the only page remaining was creating one for the home page. Changes made to index script & googleService script has been reflected accordingly.

home



Authentication

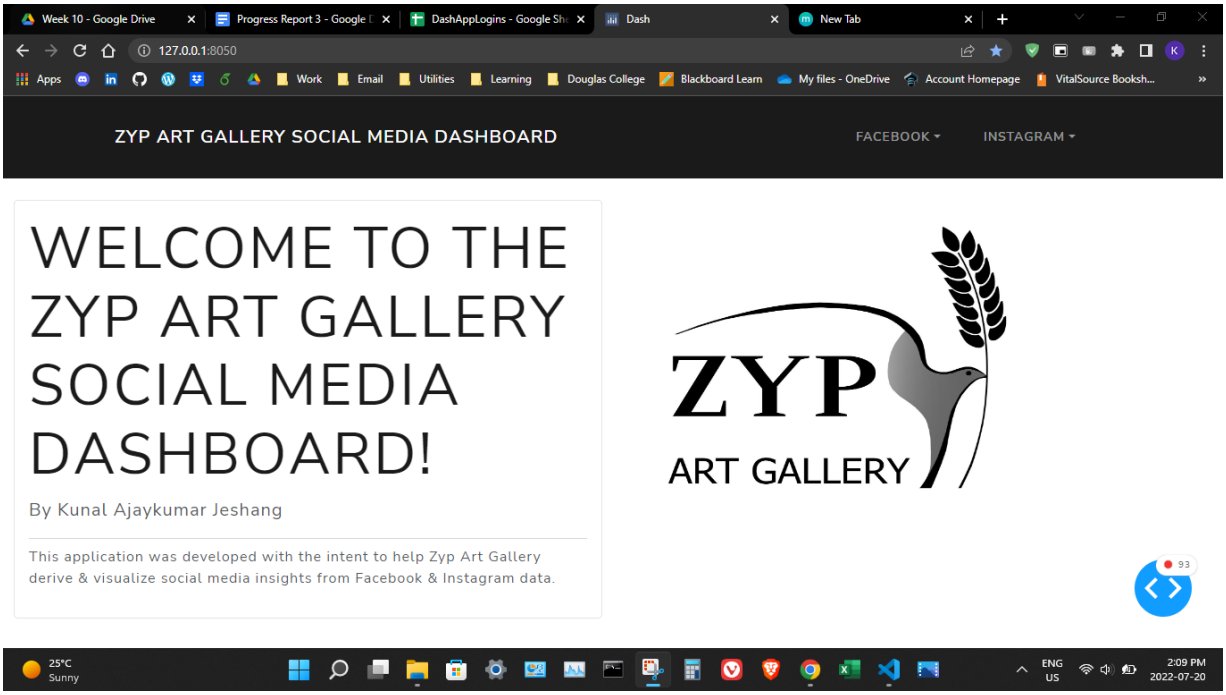




As can be seen from the above screenshot, upon running the application, the login pop-up would appear. To sign in, one must put in the username & password, and then select the ‘Sign

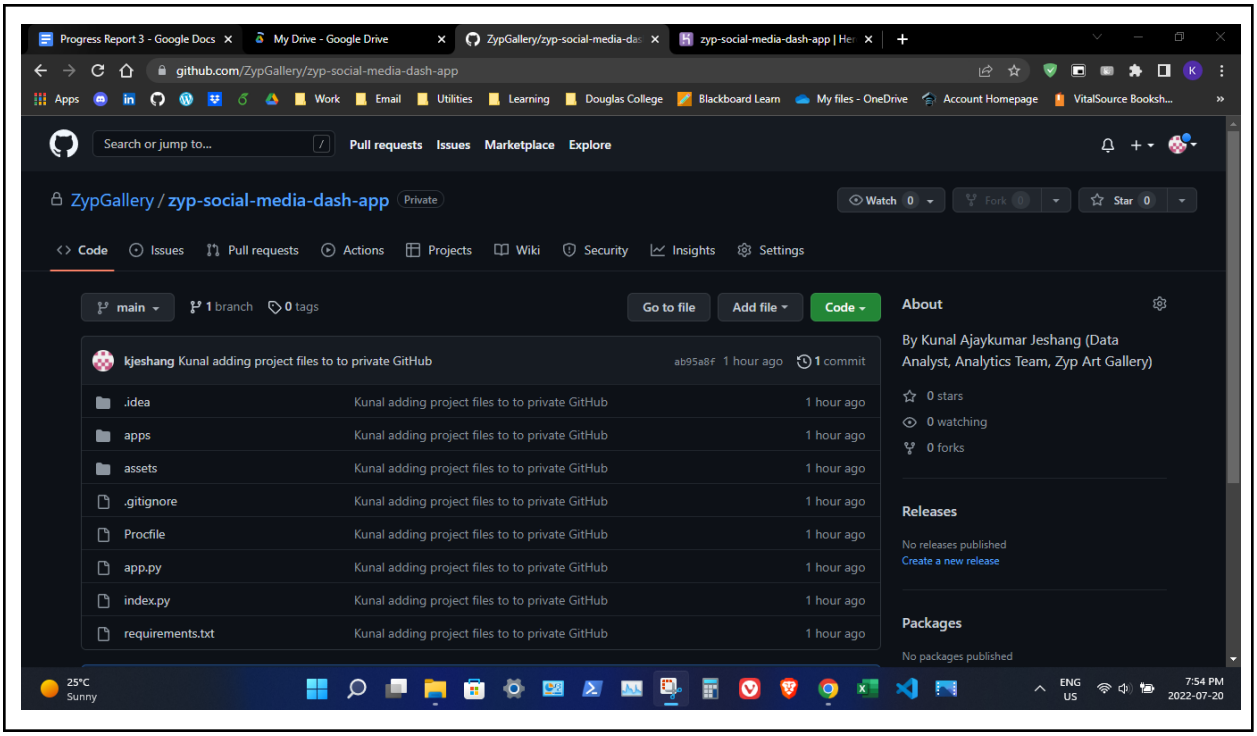
In’ button. The correct credentials are required otherwise it would ask the user again. If the ‘Cancel’ button is selected, then a blank page would be seen with the words “Login Required”.

Home page



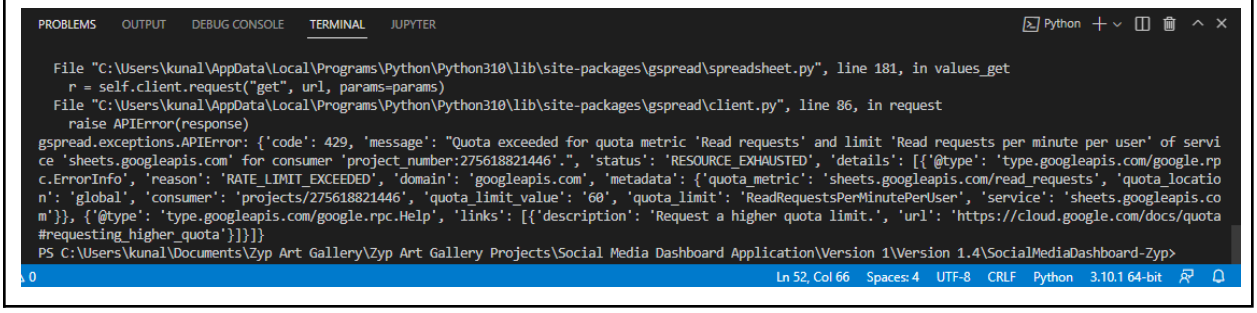
After sucessful login, this is the home/landing page.

Proof of project uploaded to organization GitHub



The above screenshot may not refer to how the actual application looks, but shows proof that I uploaded the project to the organization's private GitHub.

Proof of Google Sheets API threshold error

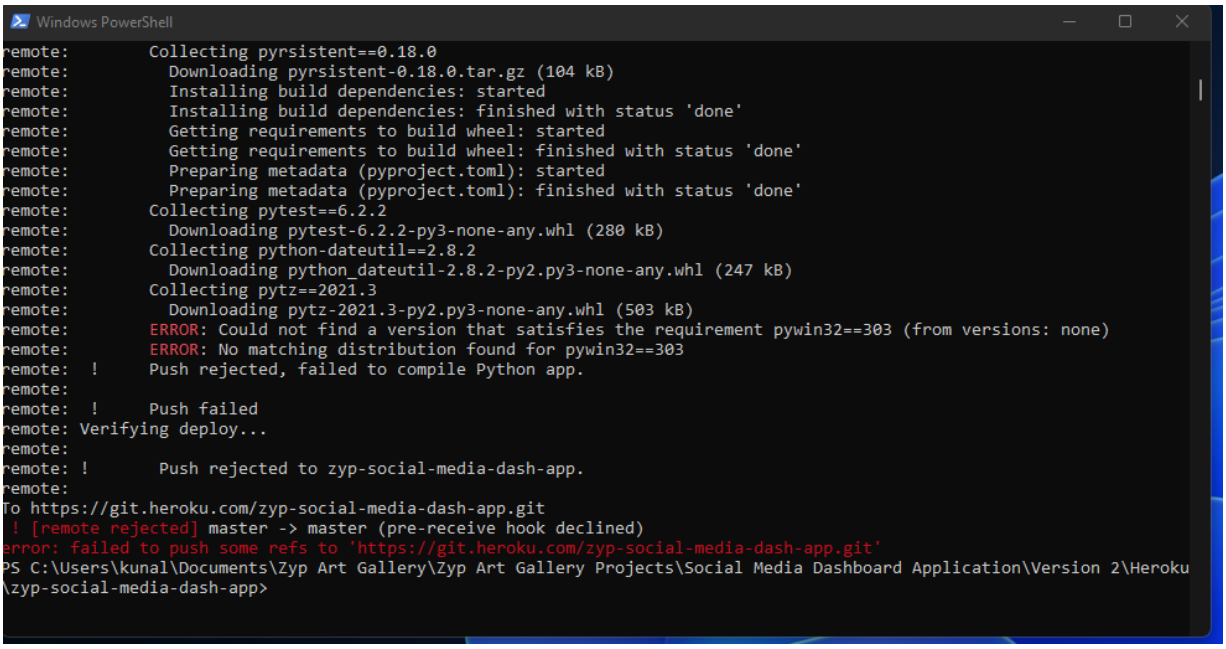


The above screenshot may not refer to how the actual application looks, but shows proof that of the Google Sheets API request threshold limit error which stops the application from running if at has been started-and-stopped too many times, in turn, importing data from Google Drive consequently too much.

Deployment Attempt Screenshots

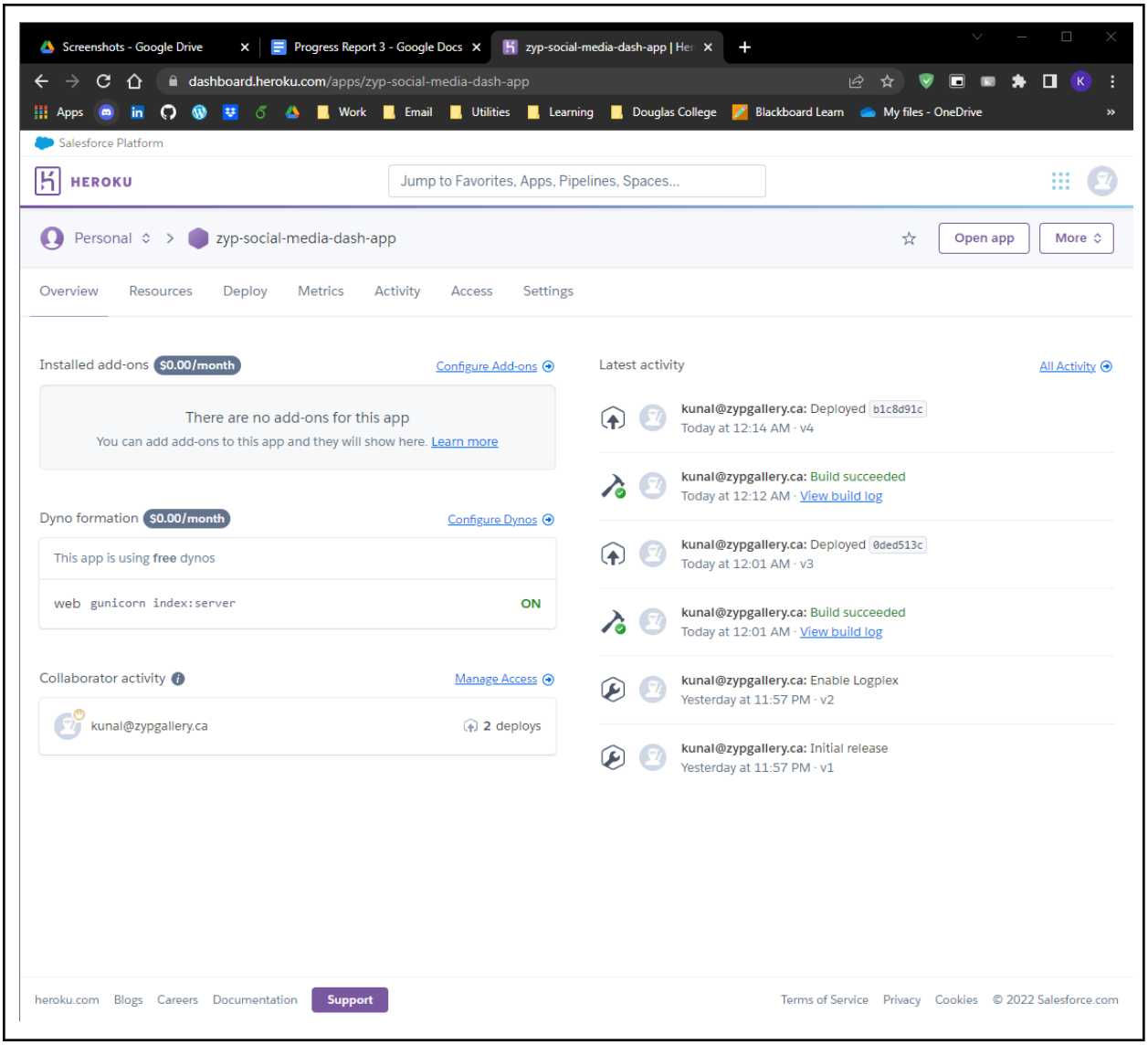
This section contains screenshots showing proof of my attempts to try to deploy the application on Heroku, along with brief descriptions of the situation.

Proof of deployment failure



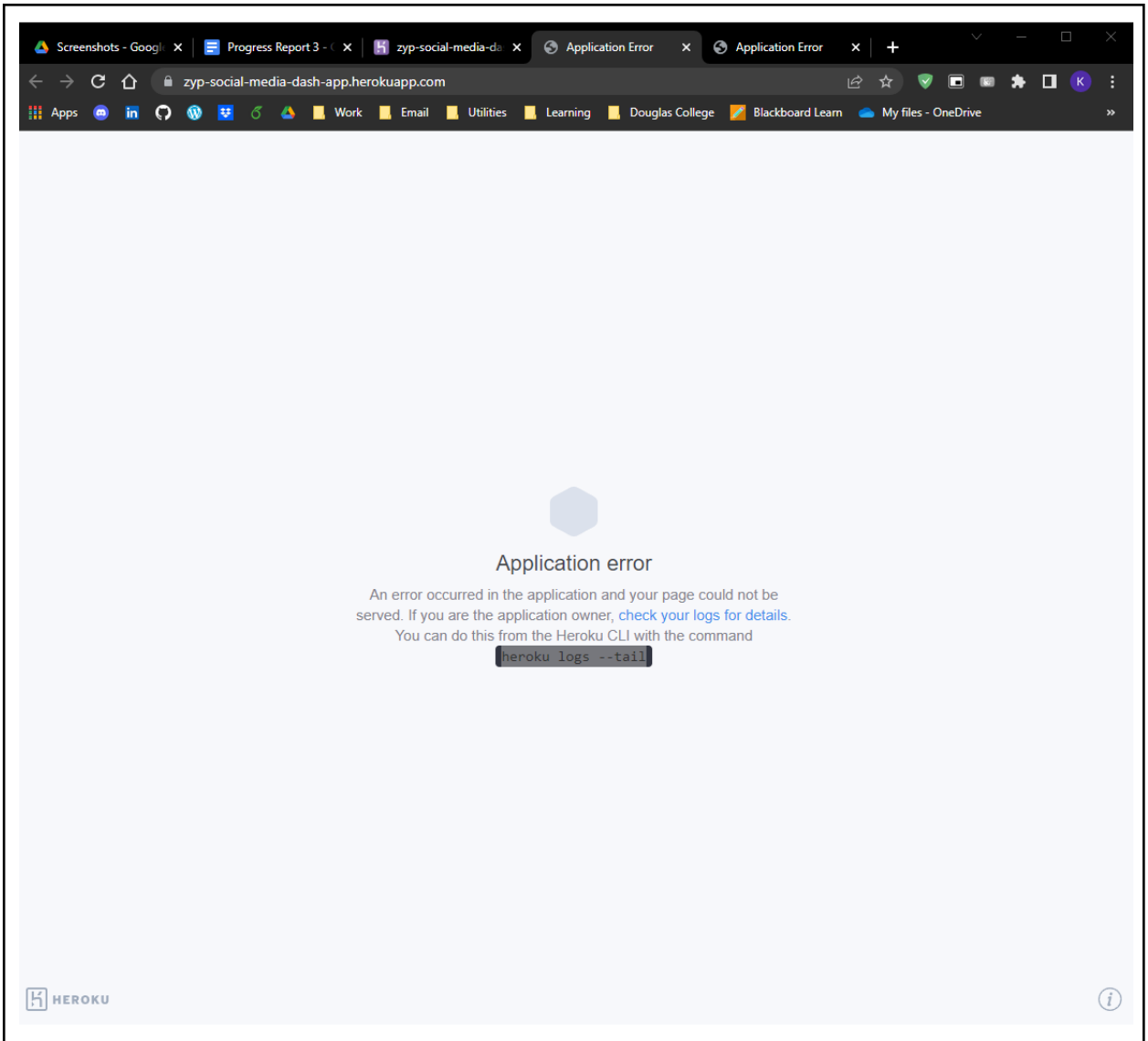
The above screenshot shows an error I received the first time I tried to deploy my application to Heroku. What occurred here is that the version of the project I tried to deploy did not have a virtual environment. Thus, my requirements text document had a long list of dependencies which had a ‘pywin32’ package. After doing some research, the aforementioned package is known for causing Heroku deployment issues. Thus, in this attempt deployment did not even succeed.

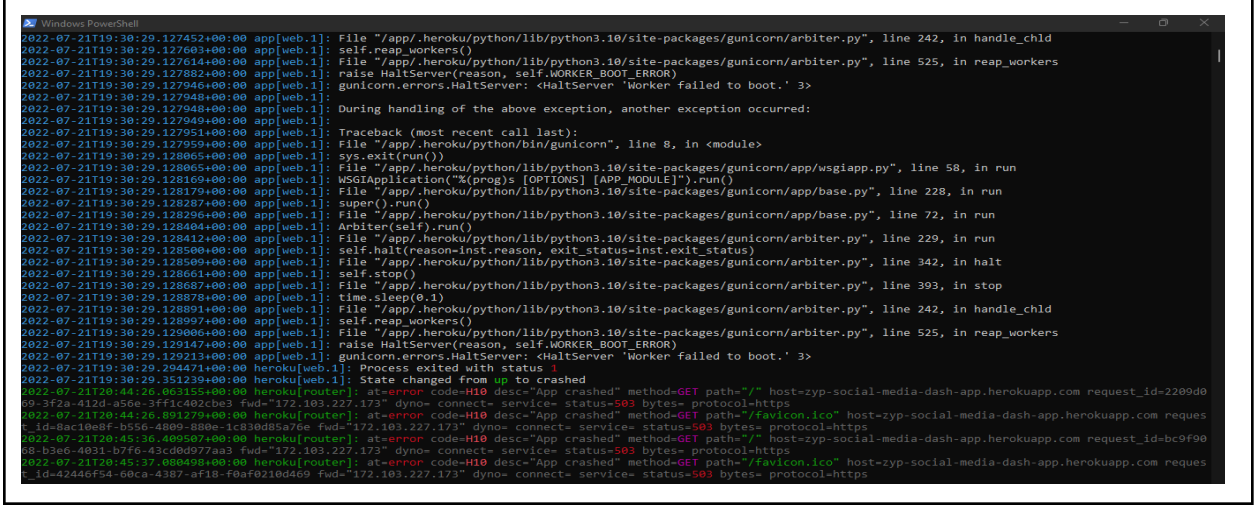
Proof of successful deployment



The above screenshot is proof that I eventually managed to successfully deploy the application by following the apporpriate steps.

Proof of Application error


Application Error
Application Error
zyp-social-media-dash-app.herokuapp.com
Application error
An error occurred in the application and your page could not be served. If you are the application owner, check your logs for details.
You can do this from the Heroku CLI with the command
heroku logs --tail
HEROKU


Windows PowerShell
2022-07-21T19:30:29.127452+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 242, in handle_chld
2022-07-21T19:30:29.127603+00:00 app[web.1]: self.reap_workers()
2022-07-21T19:30:29.127614+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 525, in reap_workers
2022-07-21T19:30:29.127882+00:00 app[web.1]: raise HaltServer(reason, self.WORKER_BOOT_ERROR)
2022-07-21T19:30:29.127946+00:00 app[web.1]: gunicorn.errors.HaltServer: <HaltServer 'Worker failed to boot.' 3>
2022-07-21T19:30:29.127948+00:00 app[web.1]:
2022-07-21T19:30:29.127948+00:00 app[web.1]: During handling of the above exception, another exception occurred:
2022-07-21T19:30:29.127949+00:00 app[web.1]:
2022-07-21T19:30:29.127951+00:00 app[web.1]: Traceback (most recent call last):
2022-07-21T19:30:29.127959+00:00 app[web.1]: File "/app/.heroku/python/bin/gunicorn", line 8, in <module>
2022-07-21T19:30:29.128065+00:00 app[web.1]: sys.exit(run())
2022-07-21T19:30:29.128065+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/app/wsgiapp.py", line 58, in run
2022-07-21T19:30:29.128169+00:00 app[web.1]: WSGIApplication("%(prog)s [OPTIONS] [APP_MODULE]").run()
2022-07-21T19:30:29.128179+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/app/base.py", line 228, in run
2022-07-21T19:30:29.128287+00:00 app[web.1]: super().run()
2022-07-21T19:30:29.128296+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/app/base.py", line 72, in run
2022-07-21T19:30:29.128404+00:00 app[web.1]: Arbiter(self).run()
2022-07-21T19:30:29.128412+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 229, in run
2022-07-21T19:30:29.128500+00:00 app[web.1]: self.halt(reason=inst.reason, exit_status=inst.exit_status)
2022-07-21T19:30:29.128509+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 342, in halt
2022-07-21T19:30:29.128661+00:00 app[web.1]: self.stop()
2022-07-21T19:30:29.128687+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 393, in stop
2022-07-21T19:30:29.128878+00:00 app[web.1]: time.sleep(0.1)
2022-07-21T19:30:29.128891+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 242, in handle_chld
2022-07-21T19:30:29.128997+00:00 app[web.1]: self.reap_workers()
2022-07-21T19:30:29.129006+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.10/site-packages/gunicorn/arbiter.py", line 525, in reap_workers
2022-07-21T19:30:29.129147+00:00 app[web.1]: raise HaltServer(reason, self.WORKER_BOOT_ERROR)
2022-07-21T19:30:29.129213+00:00 app[web.1]: gunicorn.errors.HaltServer: <HaltServer 'Worker failed to boot.' 3>
2022-07-21T19:30:29.294471+00:00 heroku[web.1]: Process exited with status 1
2022-07-21T19:30:29.351239+00:00 heroku[web.1]: State changed from up to crashed
2022-07-21T20:44:26.063155+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/" host=zyp-social-media-dash-app.herokuapp.com request_id=2209d0
69-3f2a-412d-a56e-3ff1c402cbe3 fwd="172.103.227.173" dyno= connect= service= status=503 bytes= protocol=https
2022-07-21T20:44:26.891279+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/favicon.ico" host=zyp-social-media-dash-app.herokuapp.com reques
t_id=8ac10e8f-b556-4809-880e-1c830d85a76e fwd="172.103.227.173" dyno= connect= service= status=503 bytes= protocol=https
2022-07-21T20:45:36.409507+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/" host=zyp-social-media-dash-app.herokuapp.com request_id=bc9f90
68-b3e6-4031-b7f6-43cd0d977aa3 fwd="172.103.227.173" dyno= connect= service= status=503 bytes= protocol=https
2022-07-21T20:45:37.080498+00:00 heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/favicon.ico" host=zyp-social-media-dash-app.herokuapp.com reques
t_id=42446f54-60ca-4387-af18-f0af0210d469 fwd="172.103.227.173" dyno= connect= service= status=503 bytes= protocol=https

The screenshots above show that despite successful deployment, the application still was not rendered on Heroku. As per the instructions on the first screenshot, I used the “heroku logs --tail - -app <app name>” command to find out what errors occurred. The errors can be seen in the second screenshot.

Project Codebase

Below is the GitHub repository of the custom dashboard application itself from my personal GitHub profile. The actual Google service account private key is not provided to protect the organization’s data privacy. However, all of the Python scripts are intact and the codebase is visible to view. The current state of the repository and the commits made to it reflect the progress that was discussed in this section of the report.
https://github.com/kjeshang/ZypArtGallerySocialMediaDashboard

Pending Implementations

- Deployment (may not be feasible at this current time)
    - Research
    - Practice and testing

Proposed Revisions

As discussed in the respective implementation progress section, there has been certain degree of challenges in regards to deployment of the application via Heroku. As a lot of time has elapsed throughout the term to develop the dash application, there may not be enough time to invest in researching and testing deployment. However, I will try my best to figure out why the Heroku errors if time permits. Furthermore, the remainder of time may be needed to complete the final term report and any other related deliverables. For now, I would have to re-evaluate how the application can be used at this current time. At its current form, non technical people may not be able to use the application due to the challenging installation of Python and the various dependencies to run the application. Thus, the application in its current form would be used as an internal tool by the Analytics team. The Analytics team members have the technical skill level required to run the application without much challenges. If they do face challenges, it would be easy for me to guide them compared to those that are part of other teams.

To create a version that is usable by members of the Analytics team, perhaps I could incorporate the social media extraction code to the dashboard application. This way, the extraction code is chained to running the application so that members of the Analytics team are using the social media data files as CSVs. Although, this is out of the scope of this course and depends on how the members of Analytics team would like to use/operate the dashboard application locally.

If I had more time, I would like to explore ways to stay within the Google Sheets API threshold limit. This would also reduce the size of the application itself whilst prospectively deploying to Heroku, as well as help the application itself run faster both locally and via Heroku. I would also like to learn if there is any possibility to export the dash application as an executable program.

Progress Report 2

Post-submission of the midterm report until the time of this section's writing (i.e., June 27th to July 11th), I resumed development of the dashboard application. I managed to complete all of the major sections of the dashboard application, which consists of all Facebook & Instagram Sections' page structure, visualizations and related interactive elements. Thus, from an analytical perspective, the dashboard application is usable to derive social media insights. The forthcoming paragraphs describe the progress made in greater detail.

Based on what was discussed in the 'Progress Report 1' section, I updated the Facebook Country section page to include a region scope filter. This allows the choropeth and bar chart to adjust based on the region scope radiobutton option selected. There is also a "World" option included to show the default view of the choropeth and bar chart, which is without a filter on any particular region. Also, I completed the Facebook Canadian City section. Initially, I was making requests directly to the GeoName API (hosted by the CIC website) to retrieve longitude and latitude data although this was prone to error from the API's end. I then resorted to downloading the data from the CIC website as a CSV then filtering the data using Excel to only include locations that are cities & towns. Processing the data for the Facebook Canadian City section already took long due to the extracted social media data have 9076 columns. Retrieving the longitude and latitude data via a CSV helped reduce the overhead on my computer when it came to importing the Canadian City data files from Google Drive and setting up a pandas dataframe containing the location, metric value count, latitude, longitude, and province name. Although, the Facebook Canadian City section still takes approximately 1 to 2 minutes to fully render upon opening the page via the navigation bar and/or changing a filtration option on the page itself. Although, after the page is rendered, which is the visualization/s based on the filtration options, the bubble map & bar chart visualizations are clear and informative. A similar issue takes place for the Instagram Canadian City section.

During this time, I managed to complete all Instagram sections of the dashboard application. It was not as challenging as I reused the codebase from the corresponding Facebook sections. Although, I had to make adjustments to accommodate for the Instagram data files' structure along with specific metric's meaning & period (i.e., Daily, Lifetime, etc.). One of those adjustments involved the Instagram Audience sections as the daterange filter was replaced with year & week filters. This is because the metrics involving the Instagram Audience insights are technically as of Lifetime period per week. The year & week filters adjust the visualizations in a similar manner to that of the daterange filter. However, the visualizations of the Instagram Audience sections are nearly exact to that of the corresponding Facebook Audience sections. The only exception to this is the Instagram Time of Day section as it only contains a bar chart. The associated data file to this page does not contain viable historical data to create the area chart and line chart for the purpose of comparison. The Instagram Post section is quite different from that of the Facebook Post section, thus the page looks different as well. There is a Dash data table at the bottom of the page containing brief information about Instagram posts within the specific daterange. Upon selection of one of the rows on the Dash data table (which refers to a particular post), a card would appear on the right side of the screen containing the full

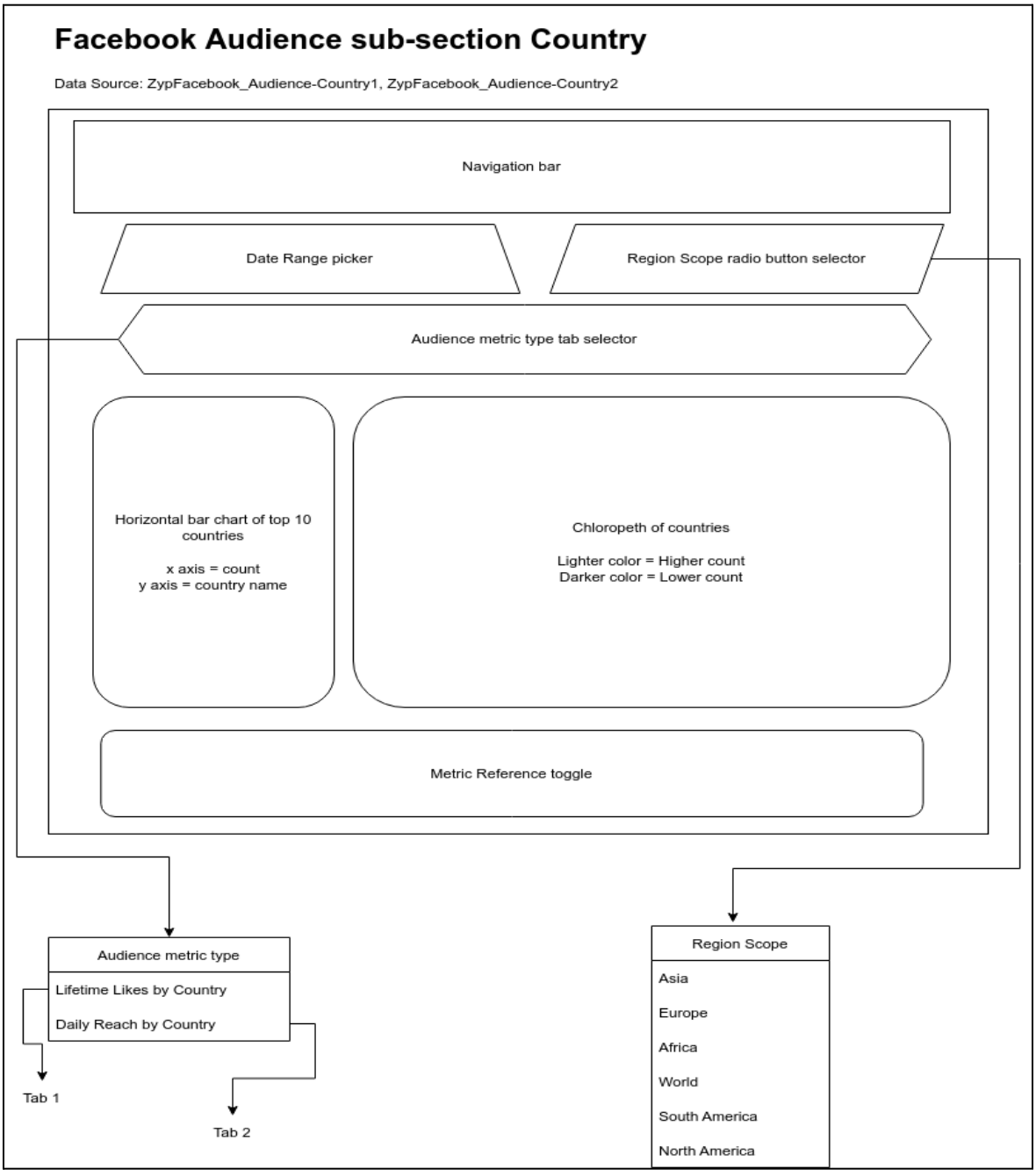
Instagram post caption, media type information, and related information. The left side of the page would show a bar graph showing the selected post's like & comment counts.

In the process of completing the Facebook and Instagram sections of the dashboard application, I discovered an error which was that the Google Drive & Google Sheets API were reaching a request threshold limit. Specifically, the error code was 409. This could have occurred due to so much rigorous stopping & running of the application during development. Specifically, this is caused by me working on some code, running the application, and then either a code error is thrown or the end result on the application is not what I had intended. In any case, each time the application is run, I am pulling data from my organization's Google Drive via the Google service account. The more times I run the application, the more I come closer to the request threshold limit. Thus, to make development move more quickly, I used more quickly, I utilized the CSV versions of the data files and pulled the data from my local machine, although a live version of the application would directly pull from my organization's Google Drive.

The following section provides evidence on majority of the progress that was described in the prior paragraphs.

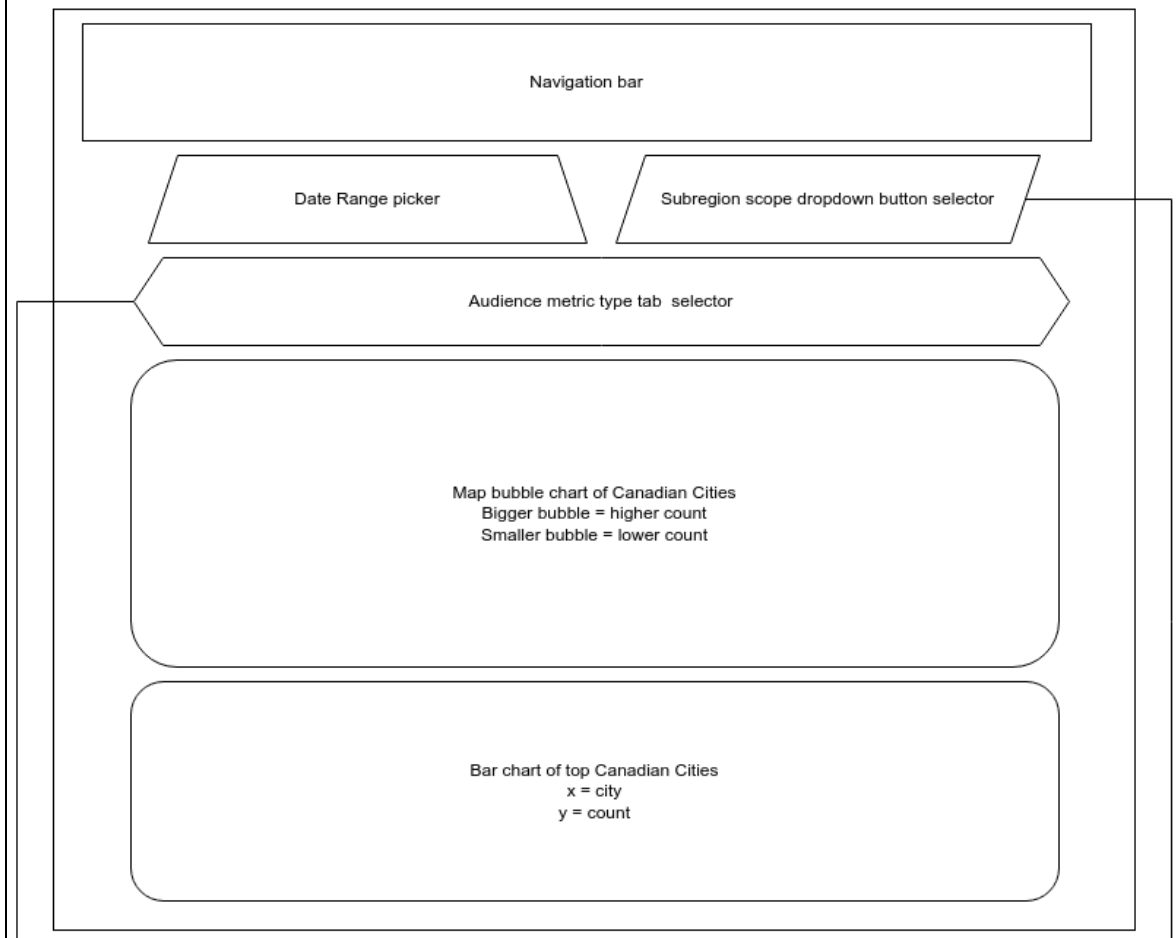
Evidence

Facebook section webpage User Interface designs



Facebook Audience sub-section Canadian City

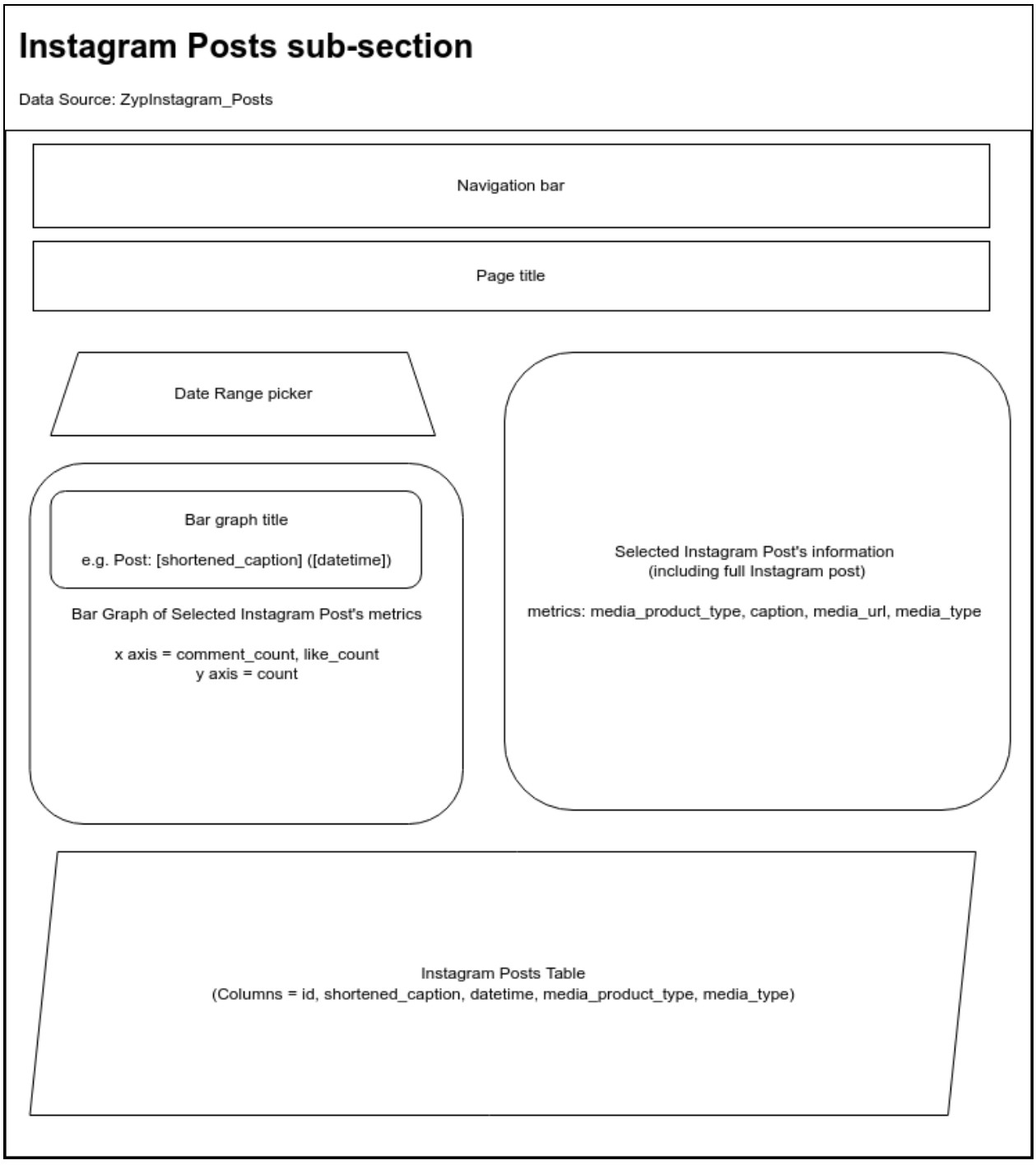
Data Source: ZypFacebook_Audience-CanadianCity1, ZypFacebook_Audience-CanadianCity2



Audience metric type
Lifetime Likes by City
Daily Reach by City

Subregion scope
Newfoundland and Labrador
Prince Edward Island
Nova Scotia
New Brunswick
Quebec
Ontario
Manitoba
Saskatchewan
Alberta
British Columbia
Yukon
Northwest Territories
Nunavut
Canada

Instagram section webpage User Interface designs

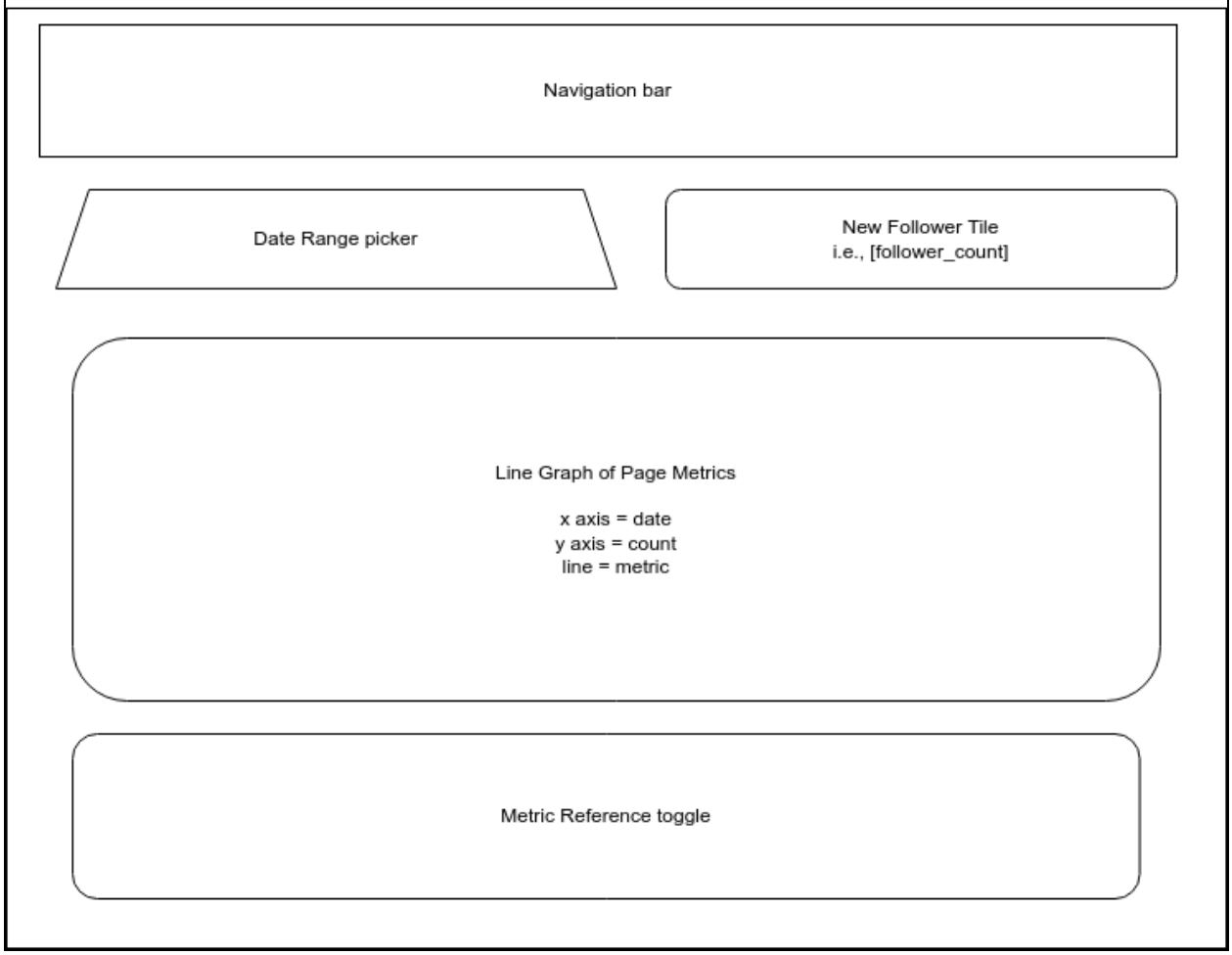


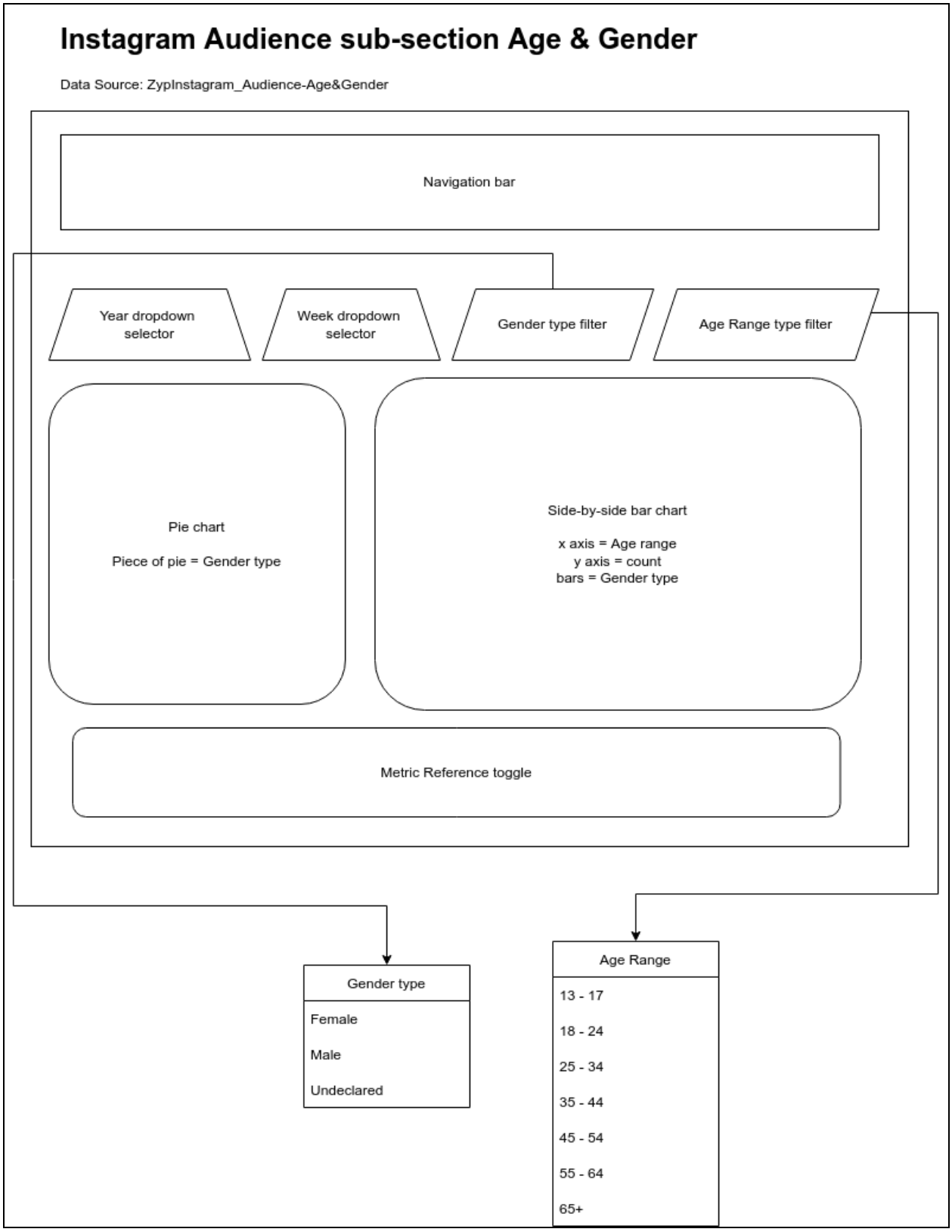
Instagram Page sub-section

Data Source:

ZypInstagram_Insights1,

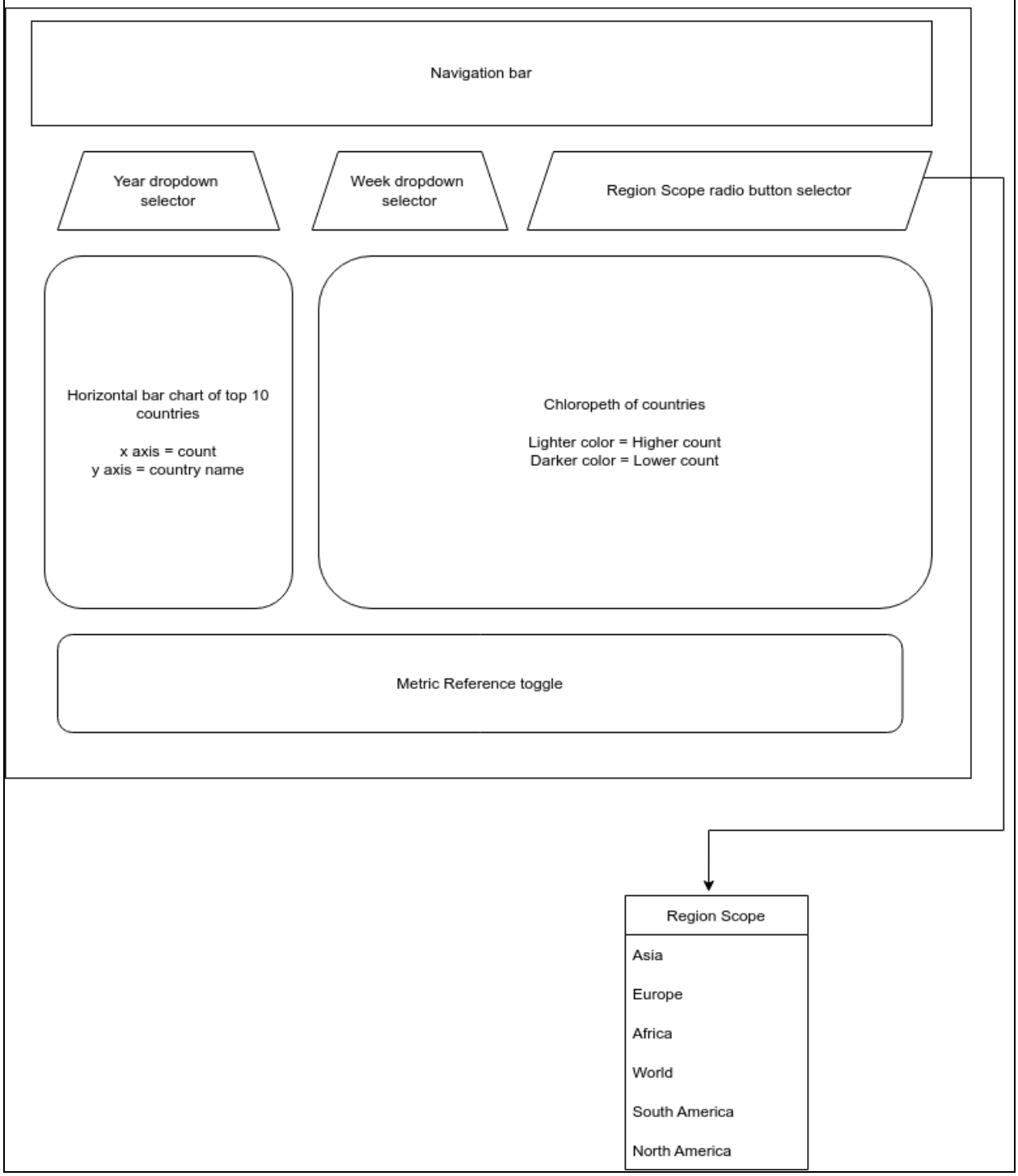
ZypInstagram_Insights2




Instagram Audience sub-section Age & Gender
Data Source: ZypInstagram_Audience-Age&Gender
Navigation bar
Year dropdown selector
Week dropdown selector
Gender type filter
Age Range type filter
Pie chart
Piece of pie = Gender type
Side-by-side bar chart
x axis = Age range
y axis = count
bars = Gender type
Metric Reference toggle
Gender type
Female
Male
Undeclared
Age Range
13 - 17
18 - 24
25 - 34
35 - 44
45 - 54
55 - 64
65+

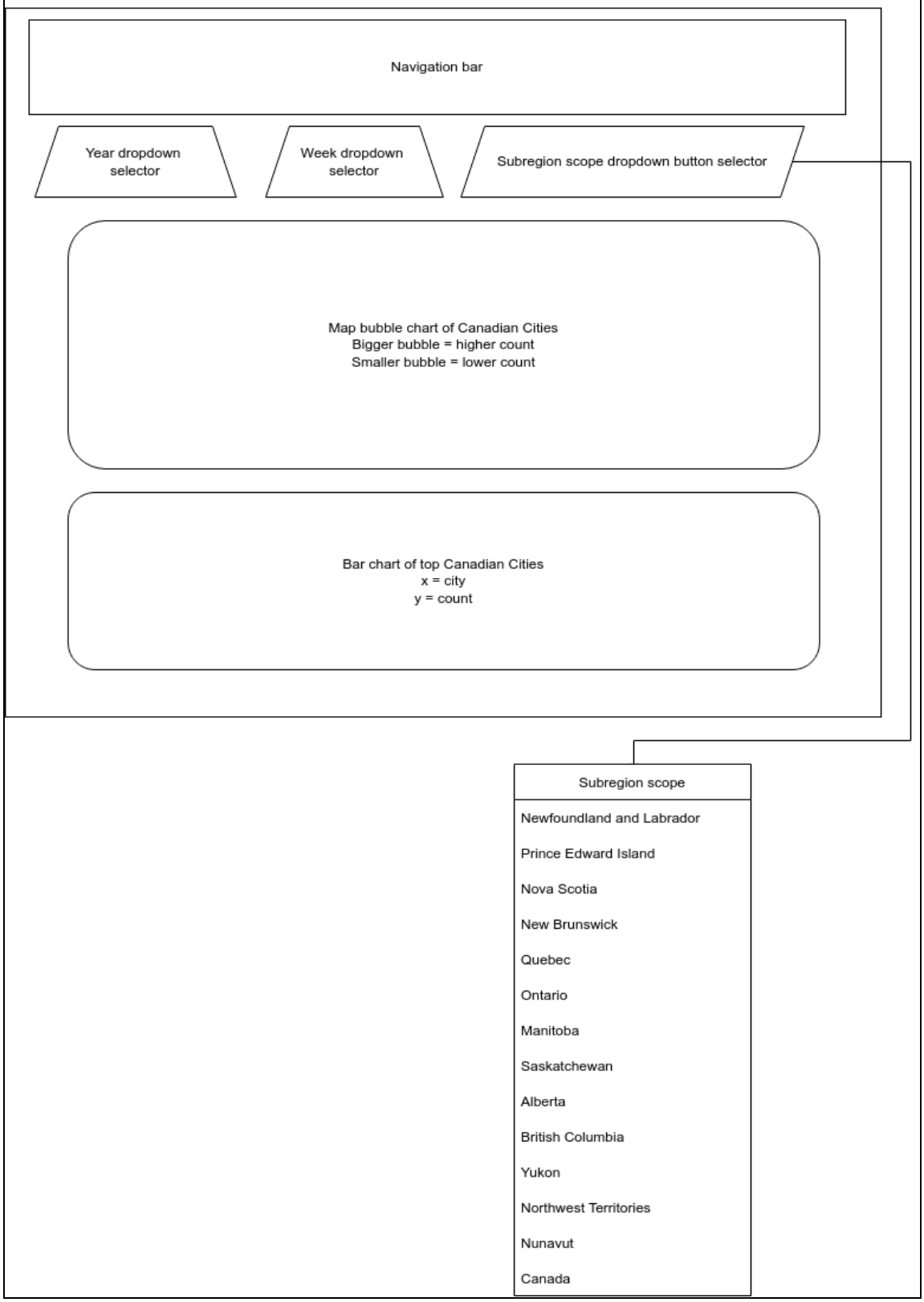
Instagram Audience sub-section Country

Data Source: ZypInstagram_Audience-Country



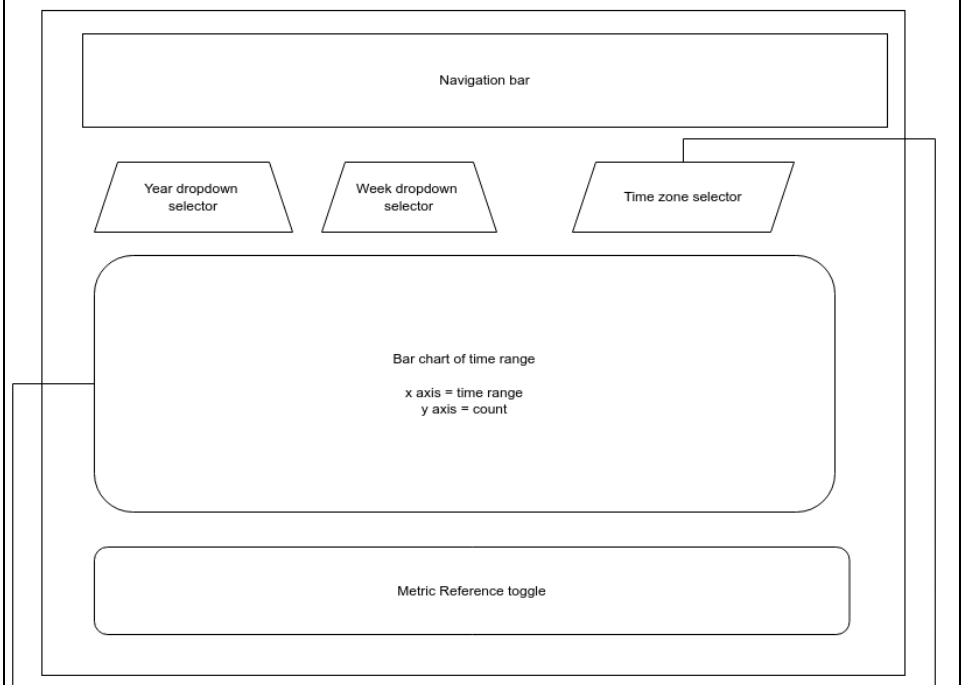
Instagram Audience sub-section Canadian City

Data Source: ZypInstagram_Audience-CanadianCity



Instagram Audience sub-section Time of Day

Data Source: ZypInstagram_Audience-TimeOfDay



Time Zone
Mountain Time
Pacific Standard Time

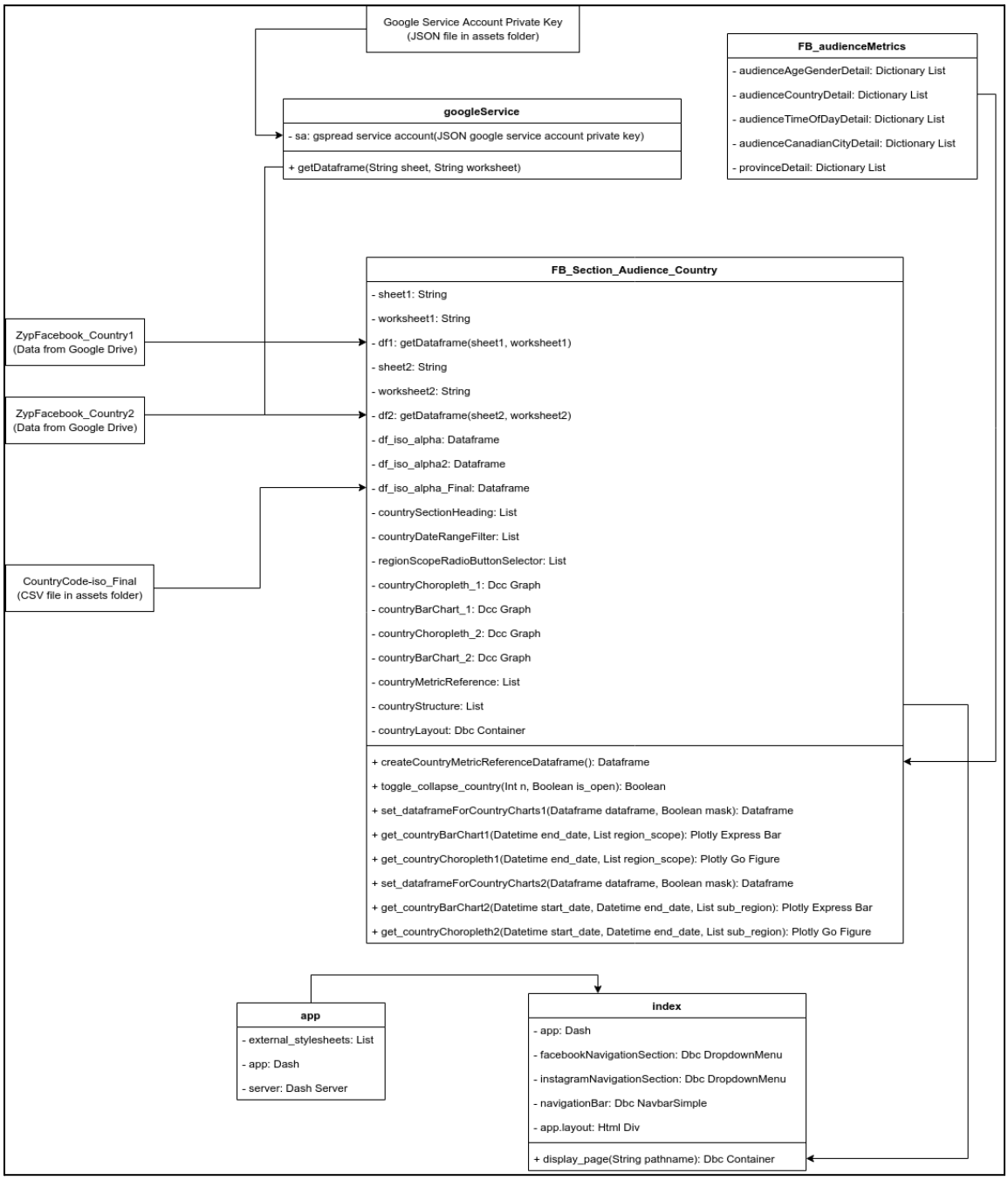
Time Range
0:00 - 1:00
1:00 - 2:00
2:00 - 3:00
3:00 - 4:00
4:00 - 5:00
6:00 - 7:00
7:00 - 8:00
8:00 - 9:00
9:00 - 10:00
10:00 - 11:00
11:00 - 12:00
13:00 - 14:00
14:00 - 15:00
15:00 - 16:00
16:00 - 17:00
17:00 - 18:00
18:00 - 19:00
19:00 - 20:00
20:00 - 21:00
21:00 - 22:00
22:00 - 23:00
23:00 - 24:00

UML

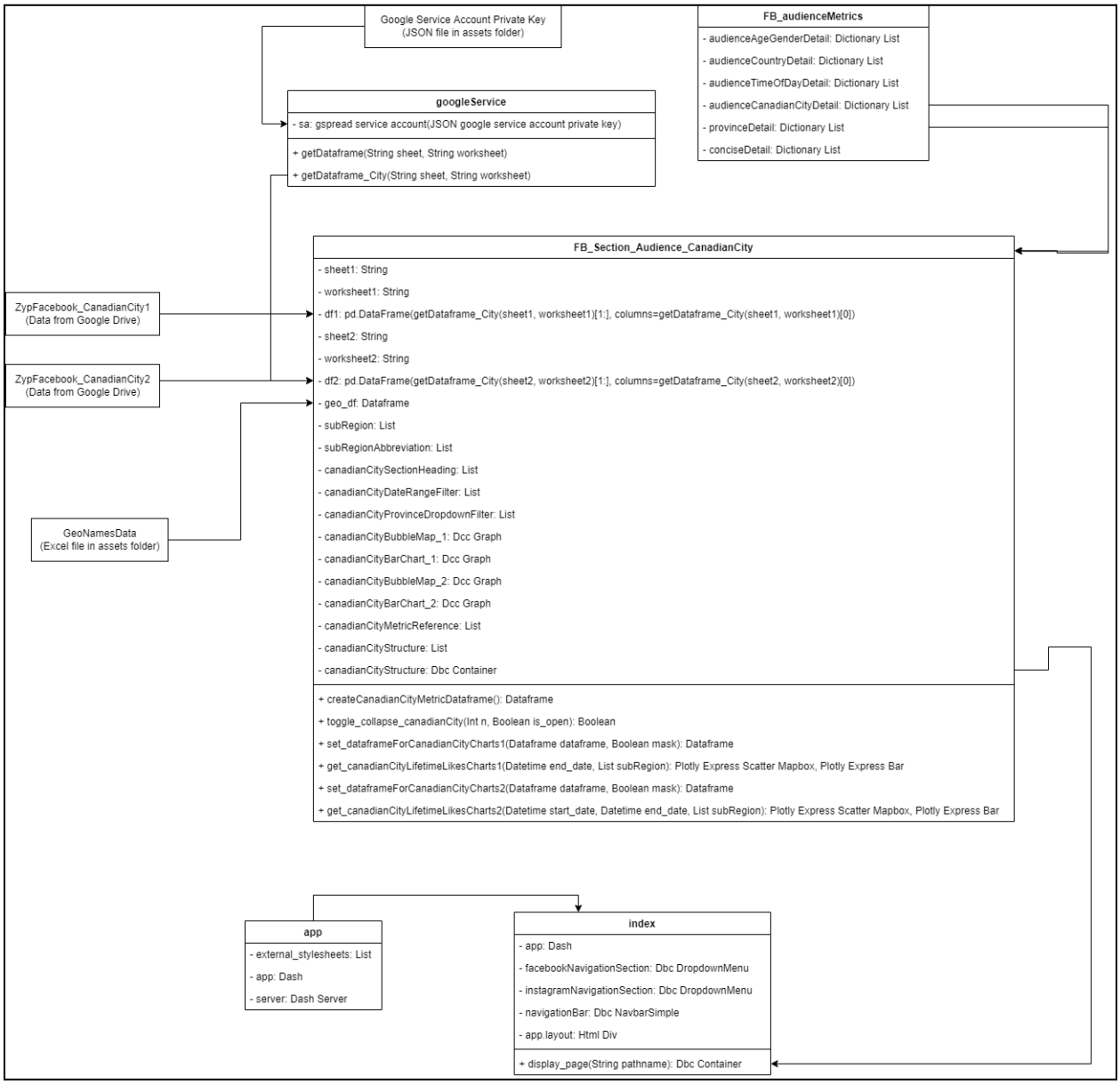
Due to the scope of the project, only the class diagrams of the remaining Facebook & Instagram sections are shown. Other diagrams such as the package diagram would not be practical. Please note that UML class diagrams of the earlier completed Facebook section pages were provided in the midterm report. Other types of UML diagrams were also provided in the midterm report.

Facebook UML Class Diagrams

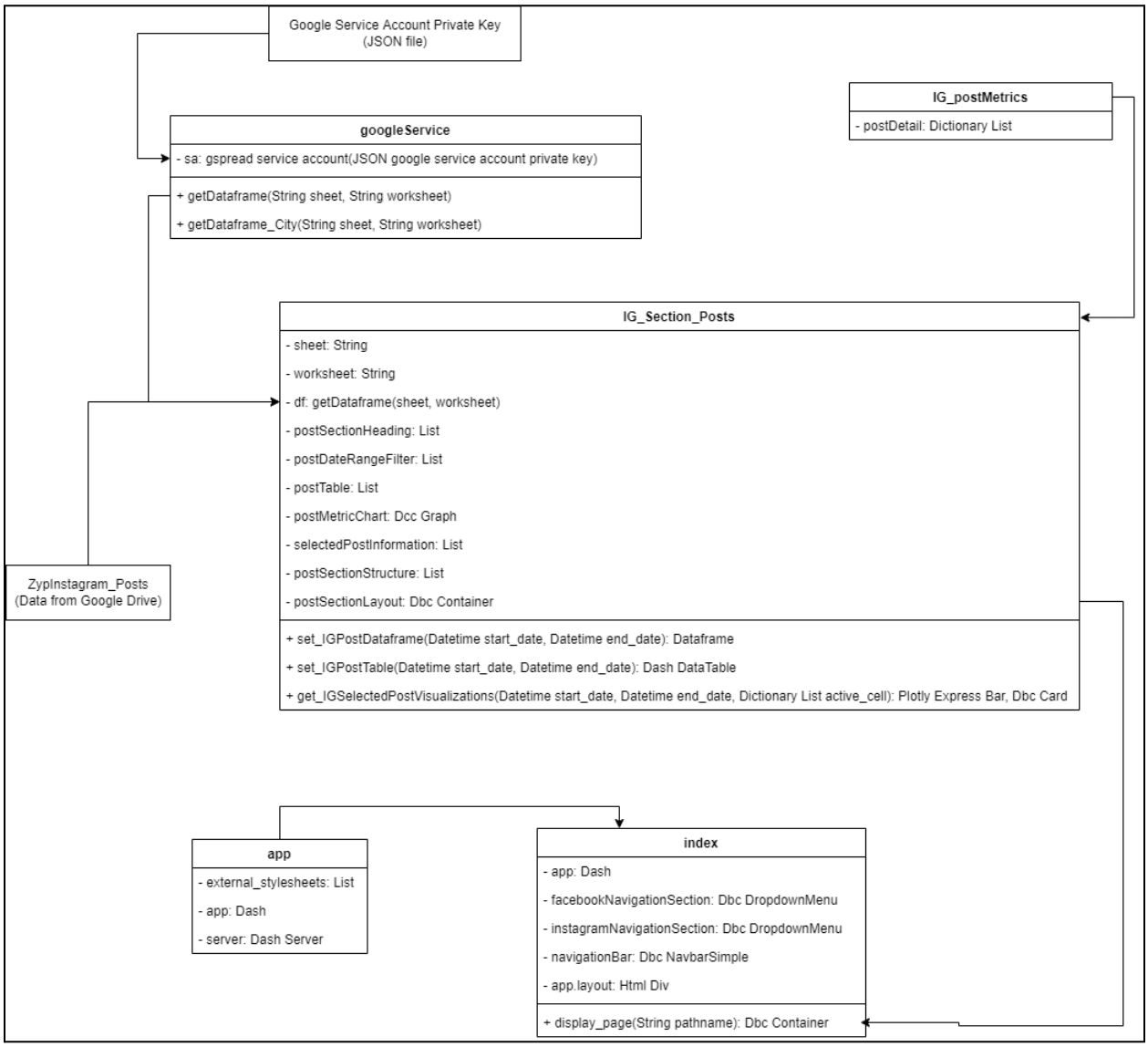
FB_Section_Audience_Country



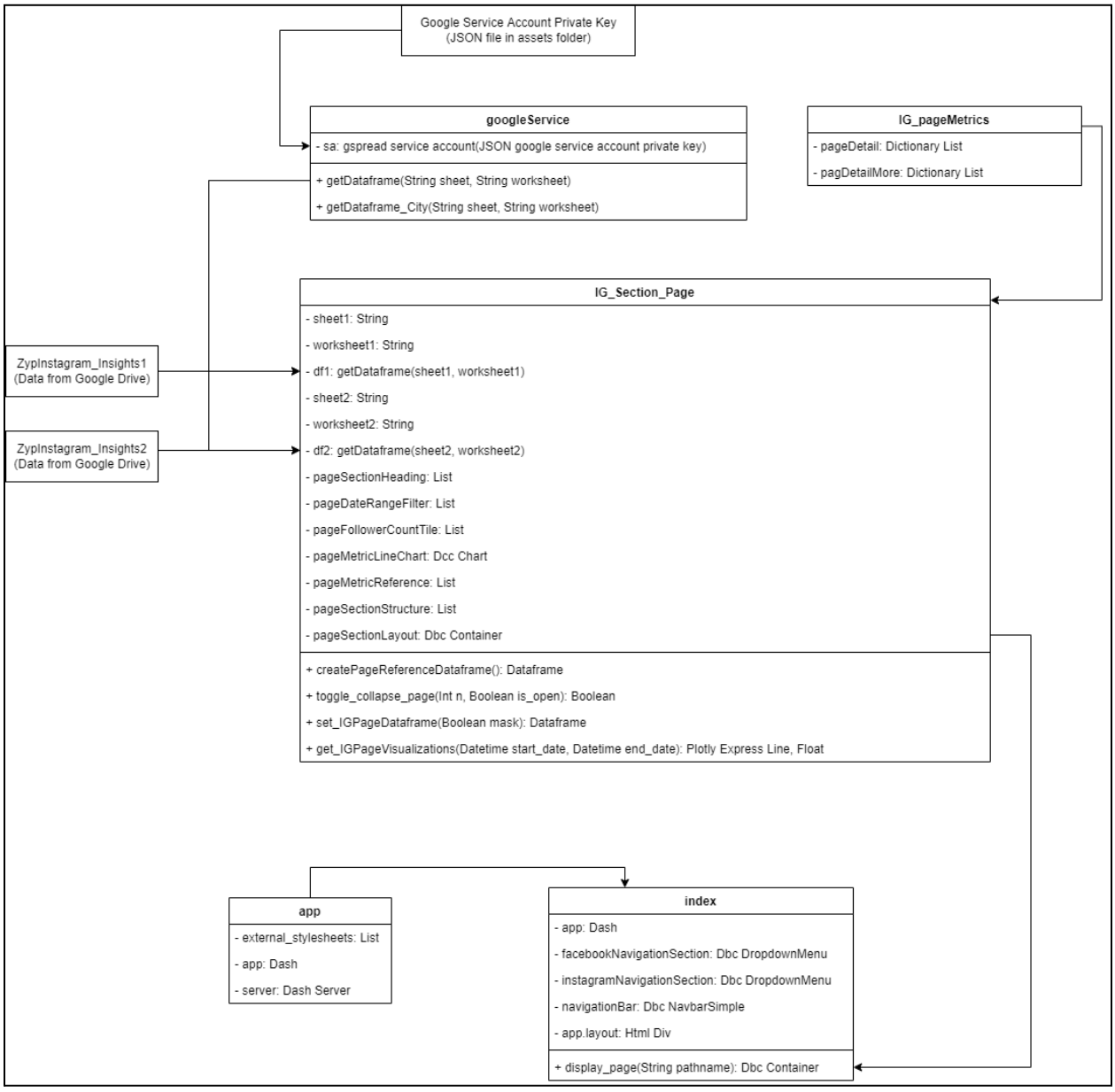
FB_Section_Audience_CanadianCity



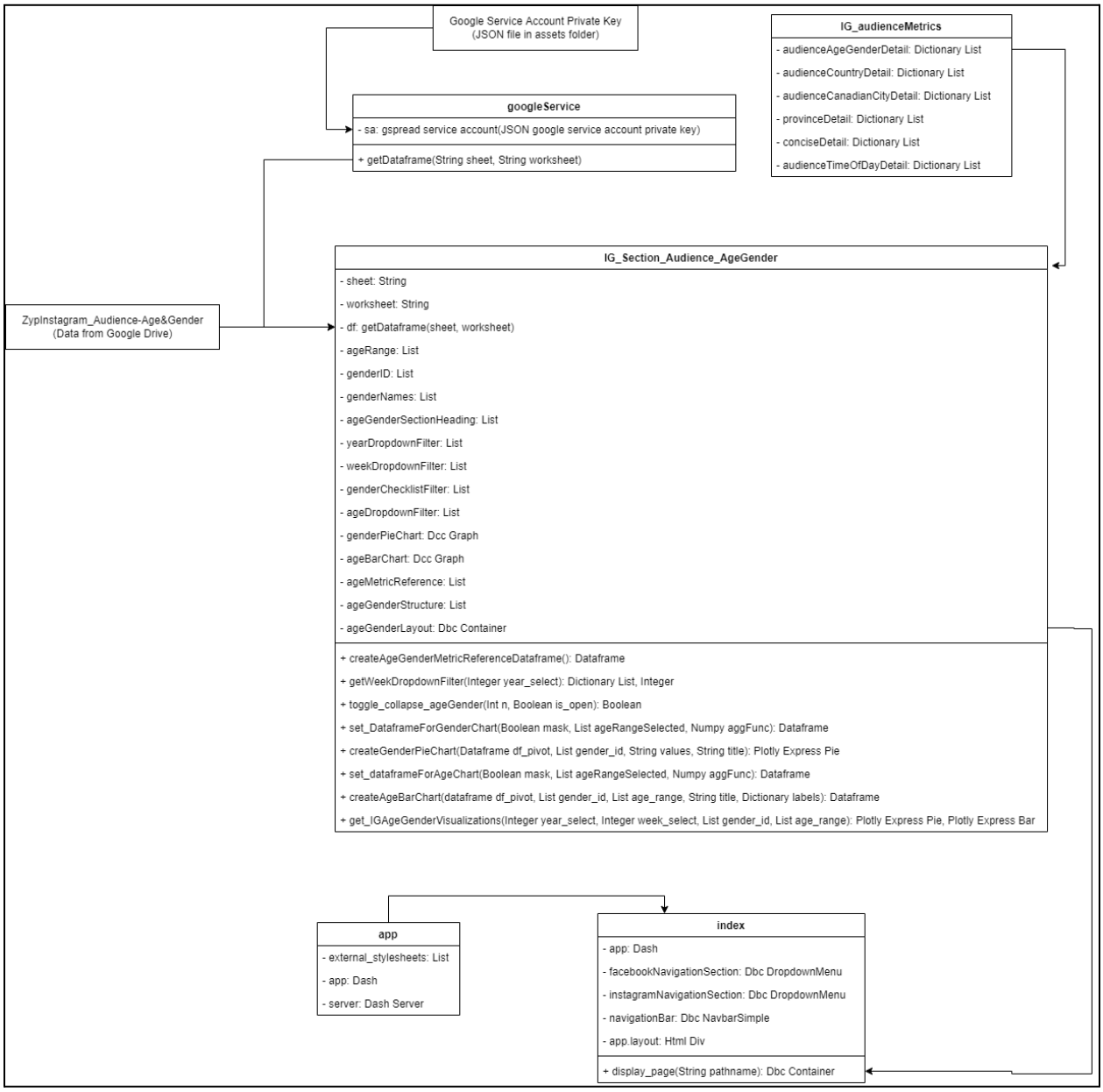
IG_Section_Posts



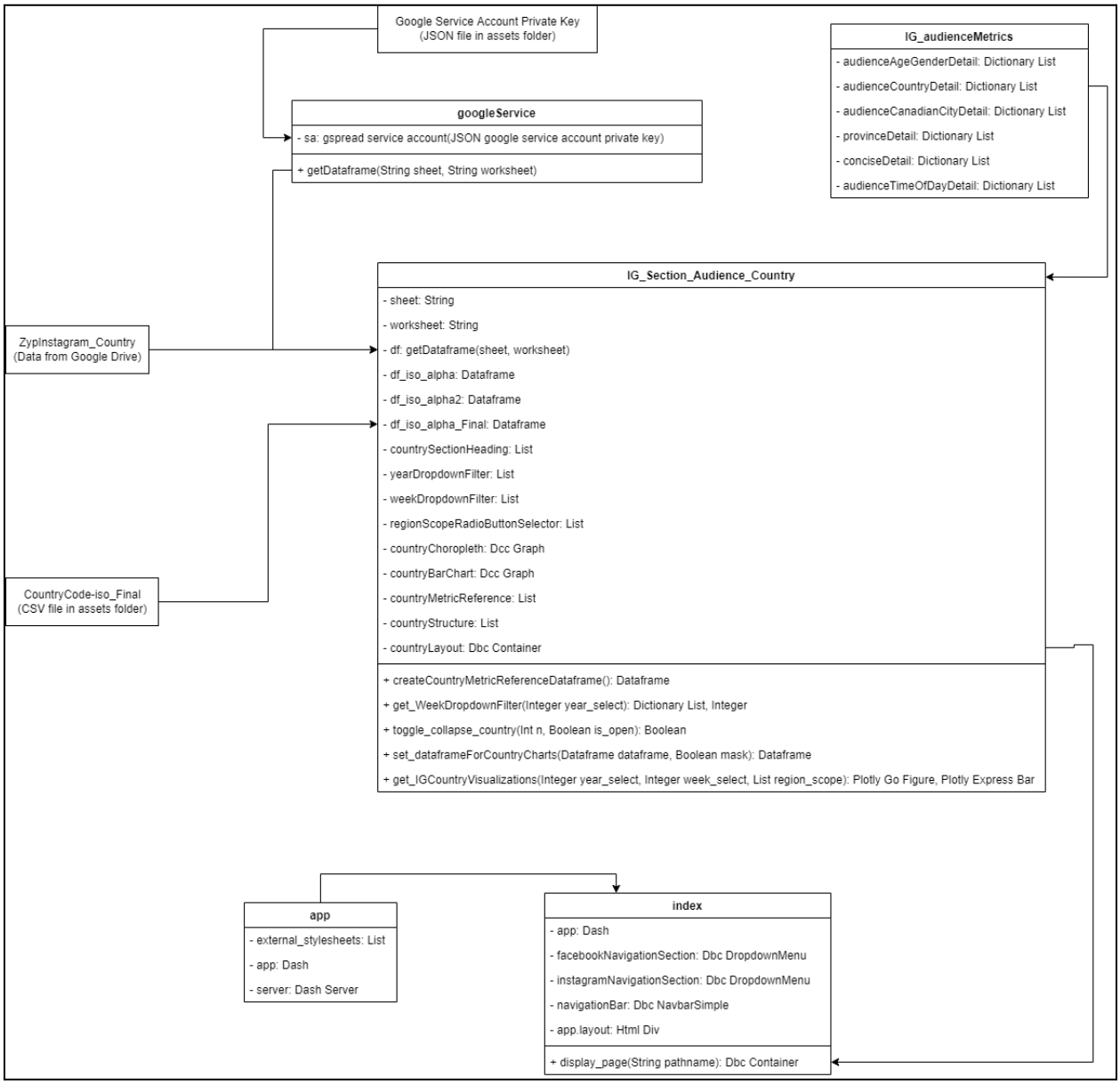
IG_Section_Page



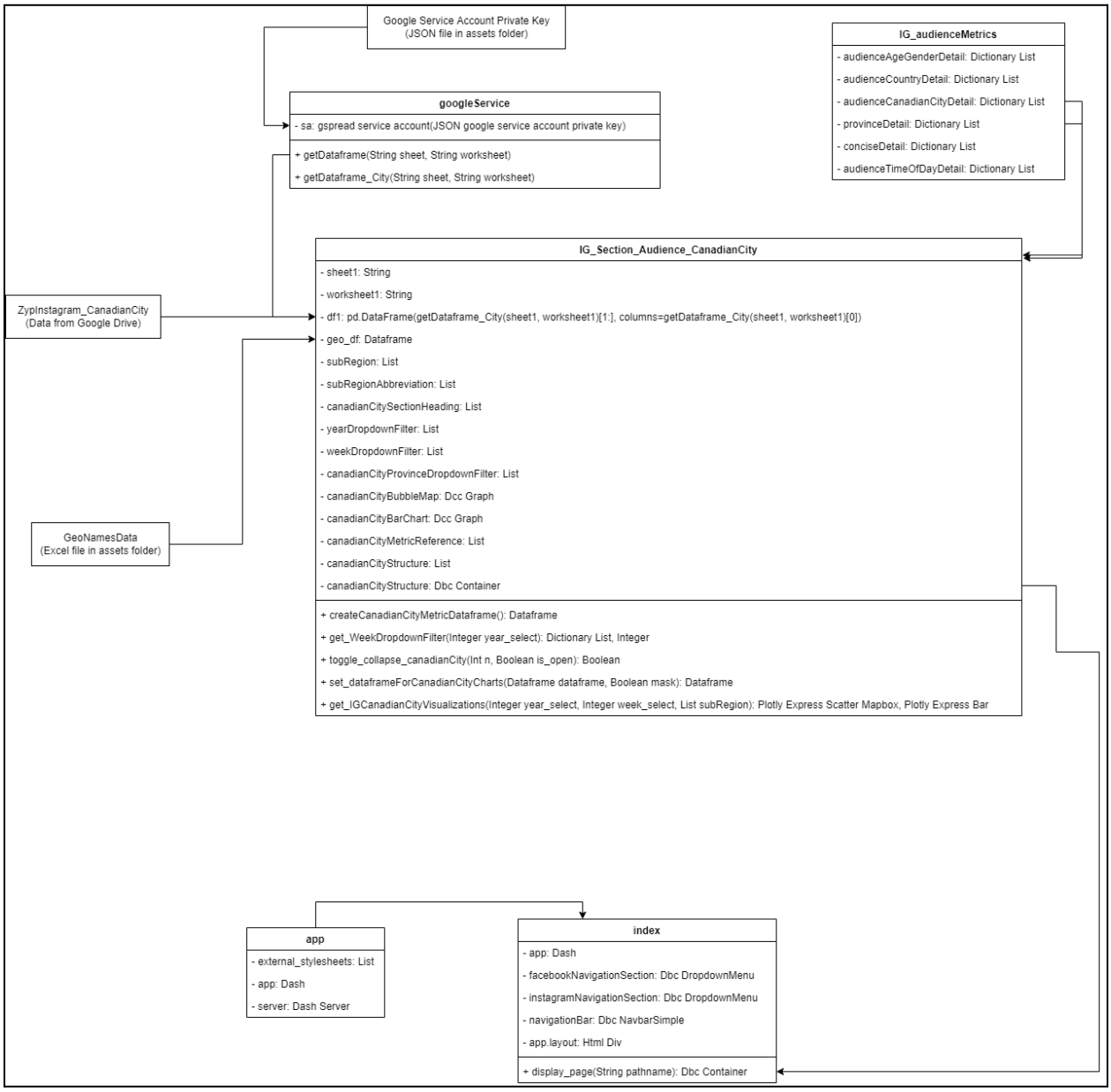
IG_Section_Audience_AgeGender



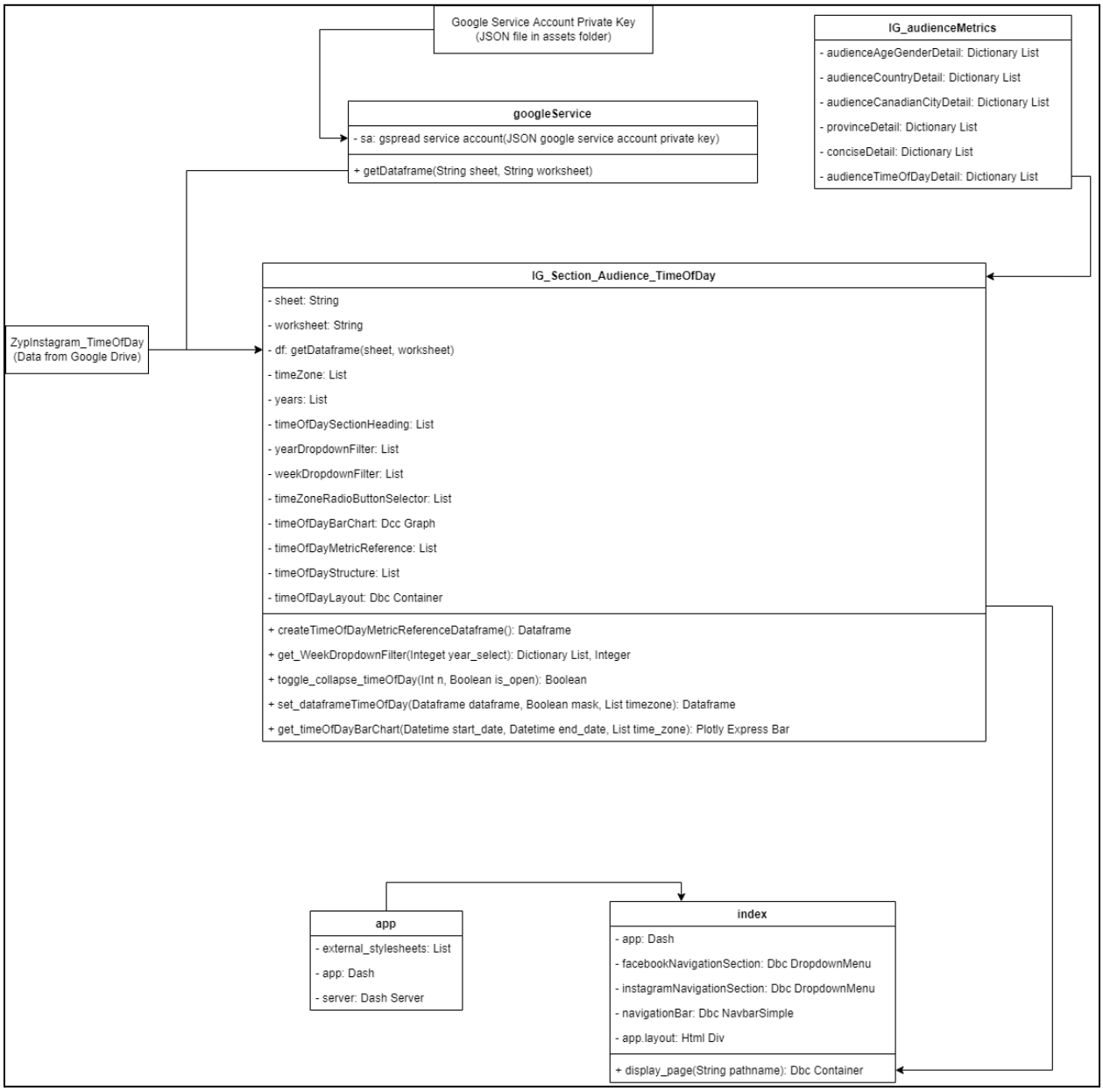
IG_Section_Audience_Country



IG_Section_CanadianCity



IG_Section_TimeOfDay



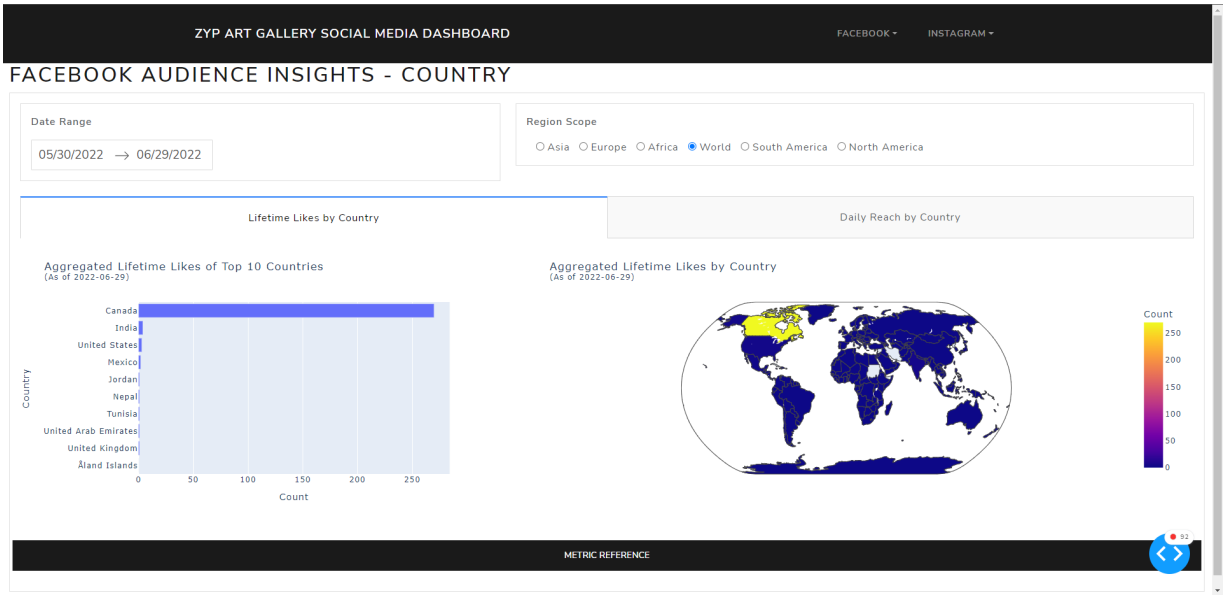
Application Screenshots

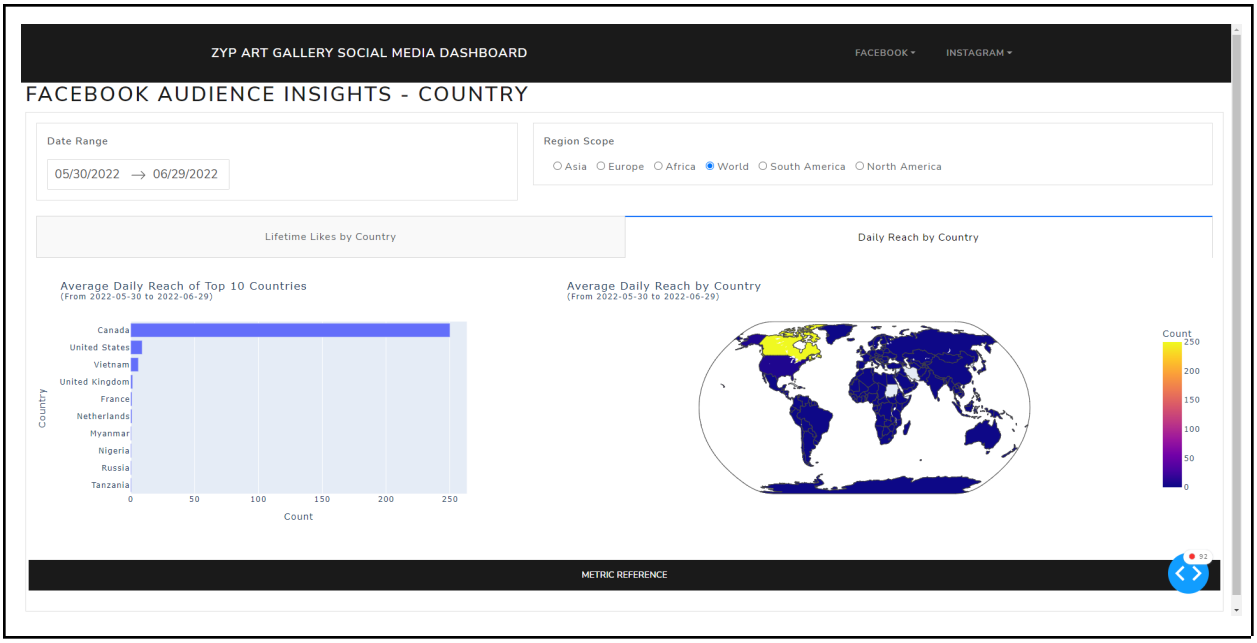
As all of the Instagram sections have now been completed, so to has the navigation bar and its various navlink options. The dashboard application has a structure that is described below.

- Title of Dashboard Application: Zyp Art Gallery Social Media Dashboard (title on navigation bar and is currently hyperlinked to the index page)
- Facebook section navigation dropdown menu (navlink options available on hover)
    - Posts
    - Page
    - Audience (this is not a link but a header within the navigation bar's dropdown menu)

        - Age & Gender
        - Country
        - Canadian City
        - Time of Day
- Instagram section navigation dropdown menu (navlink options available on hover)
    - Posts
    - Page
    - Audience (this is not a link but a header within the navigation bar's dropdown menu)
        - Age & Gender
        - Country
        - Canadian City
        - Time of Day

Facebook Audience Insights - Country



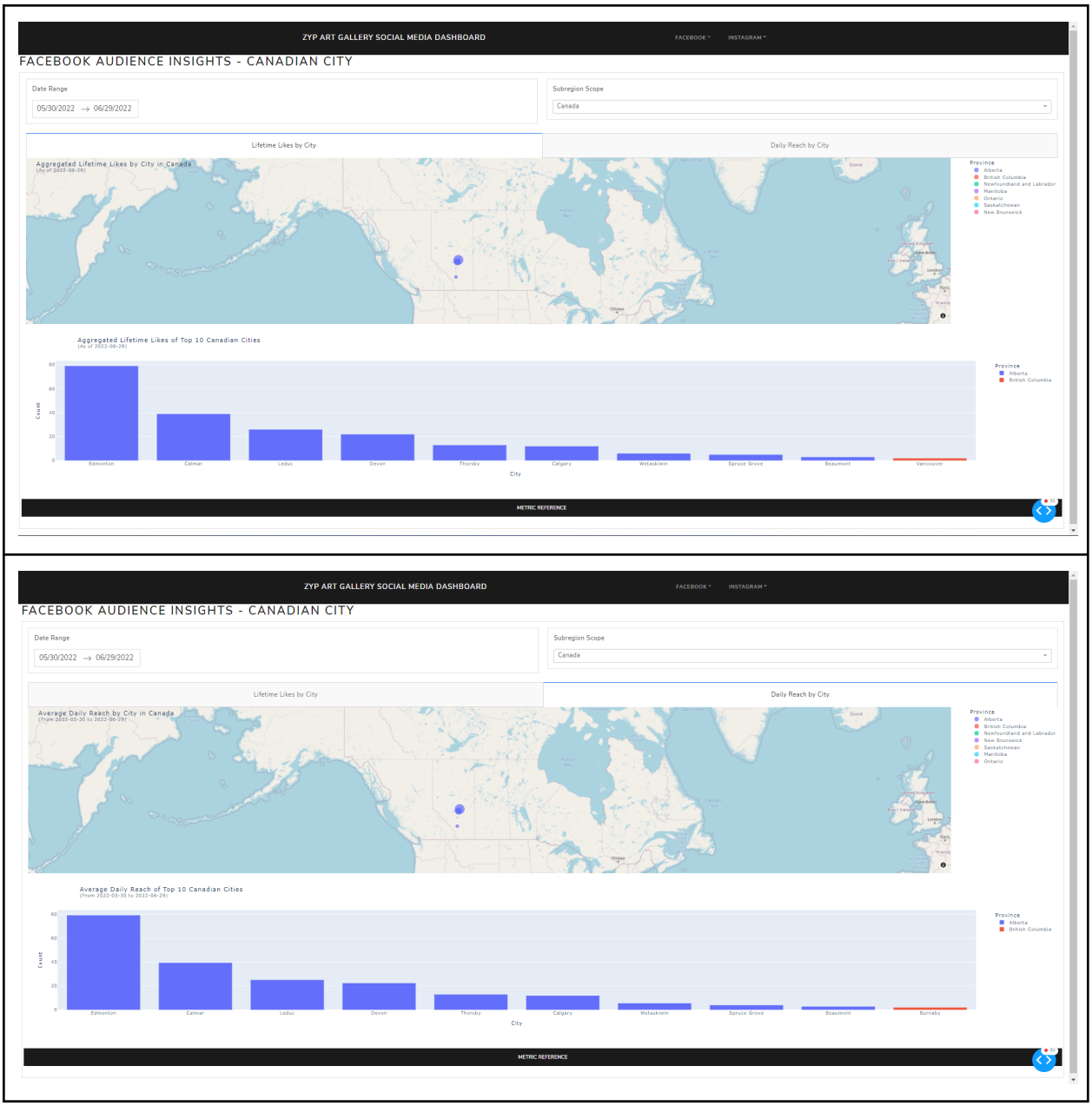


The above screenshots show the Facebook Audience Insights - Country webpage. As aforementioned in the ‘Progress Report 1’ section of this report, the intent is visualize the aggregated lifetime likes and average daily reach of the organization’s Facebook page by country.

This webpage was updated to have a region scope radiobutton list. By default, the option selected is “World”. If another region scope option is selected, the choropeth would change to show only selected region and the bar chart would adjust to show the top 10 countries based on the metric that are within the selected region. For example, if “North America” is selected, then the chorpeth would display only North America in view and zoomed in (all other region would not be in view), and the bar chart would display top 10 countries based on the metric within North America.

The data and metric information has been described in an earlier section of this report.

Facebook Audience Insights - Canadian City



The above screenshots show the Facebook Audience Insights - Canadian City webpage. The intent is visualize the aggregated lifetime likes and average daily reach of the organization's Facebook page by Canadian city.

<u>Interactive elements</u>

There are two tabs. The intent of the first tab is to communicate the likes of the Facebook page by Canadian city. The metric used in this tab is of period "Lifetime". Thus, the visualizations change based on the end date selected on the date range datepicker. The intent of the second tab is to communicate the reach of the Facebook page by Canadian city. The metric used in this tab is of period "Daily". Thus, the visualizations change based on the start & date selected on the daterange picker, and calculates the average of the metric value based on date range. The

titles of the visualizations in both tabs change based on the selections made in the datepicker. The subregion scope dropdown helps to filter the visualizations by province. However, there is also an option in the dropdown called “Canada” which is set to default. If another option selected, which is the name of a province in Canada, the visualizations would adjust to only reflect the selected province in the subregion scope dropdown.

Data

This webpage uses the “ZypFacebook_Audience-CanadianCity1” and “ZypFacebook_Audience-CanadianCity2” Google Sheets documents that are saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

The following refers to information visualized on the ‘Lifetime Likes by City’ tab.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_city City name, Province abbreviation, Country => E.g. Calgary, AB, Canada	*Lifetime Likes by City* Lifetime: Aggregated Facebook location data, sorted by city (top 50), about the people who like your Page. (Unique Users)

The following referes to information visualized on the ‘Daily Reach by City’ tab.

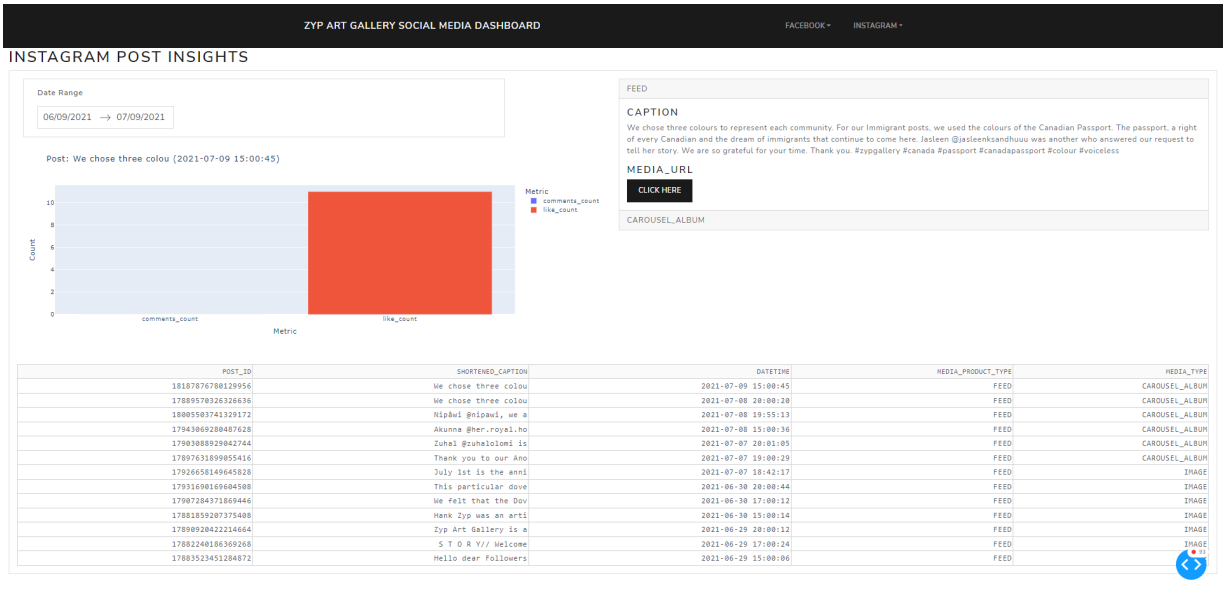
Field	***Description***
end_time	Date
Metric	***Title & Description***
page_impressions_by_city_unique City name, Province abbreviation, Country => E.g. Calgary, AB, Canada	*Daily Reach by City* Daily: Total Page Reach by user city. (Unique Users)

Visualization

Regardless of which tab is in view, there are two charts. The bar chart shows the top Canadian cities regarding the metric value. The bubble map chart shows the metric value of all Canadian cities. If there is selected option in the subregion scope that is other than “Canada”, then the bubble map chart would display bubbles referring to the metric value of Canadian cities within the selected subregion (i.e., Province). The bar chart shows the top 10 Canadian cities based on the metric value. If there is selected option in the subregion scope that is other than

“Canada”, then the bar chart chart would display bars referring to the metric value of Canadian cities within the selected subregion (i.e., Province). The ‘Lifetime Likes by Canadian City’ tab uses the “ZypFacebook_Audience-CanadianCity1” Google Sheet data file, whilst the ‘Daily Reach by CanadianCity’ tab uses the “ZypFacebook_Audience-CanadianCity2” Google Sheet data file.

Instagram Post Insights



The above screenshot shows the Instagram Post Insights webpage. The intent of this page is to communicate the performance of the organization’s Instagram posts.

<u>Interactive elements & Visualization</u>

The daterange filter on the top-left side of the triggers the display of the table of the bottom of the page. Specifically, the table displays data of the Instagram posts that within the dates selected on the daterange filter. Upon selection of a row on the table, the Instagram post’s information referring to that particular row would be displayed at the top of the page. This would consist of the comment count & like count of the Instagram post communicated in the form of a bar chart, as well as the full Instagram caption, media url, media product type, and media type in the form of a card. The media url is hyperlinked to the button, although upon clicking it a webpage opens saying that the “URL signature has expired”. A reason why the card containing the full Instagram caption is required is because the table does not render appropriately when there are column values with a very long character length. Thus, only key information is included in the table which consists of the post ID, first 20 characters of the caption, datetime of post, media product type, and media type.

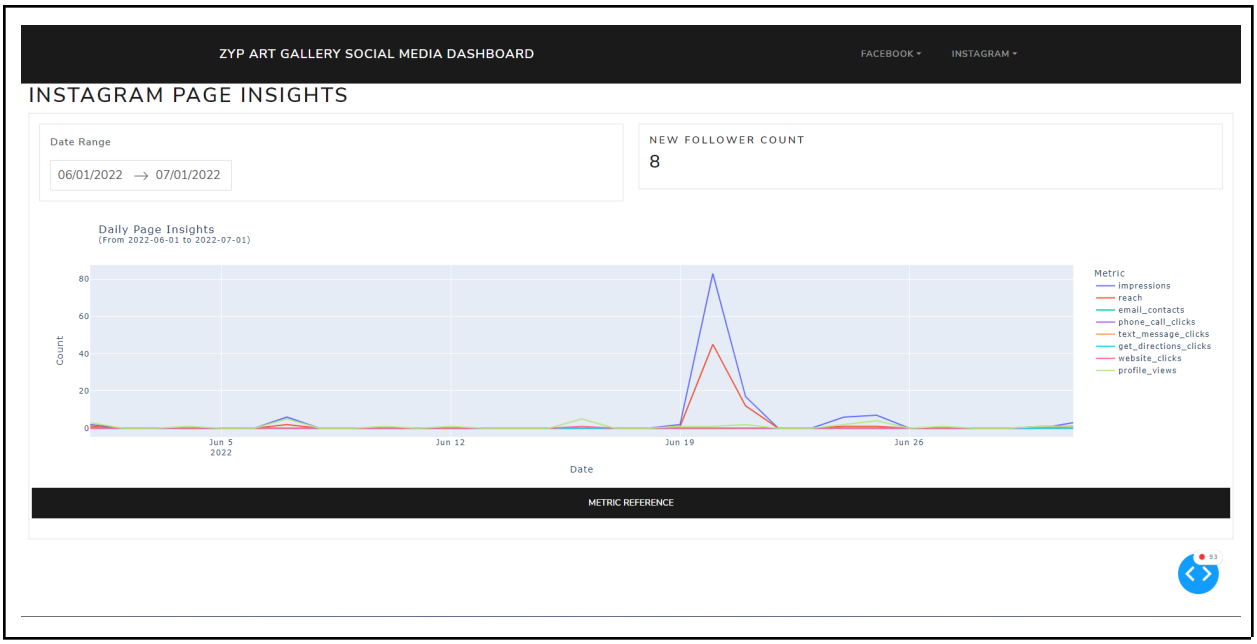
<u>Data</u>

This webpage uses the “ZypInstagram_Posts” Google Sheets document that is saved the

Organization's Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
timestamp	Timestamp of the Instagram post
id	Instagram post identification number
caption	The actual post made to Instagram
media_url	Link to the actual Instagram post
comments_count	Total count of comments received by the Instagram post
like_count	The number of likes received by the Instagram post
media_product_type	The product type of the Instagram post
media_type	The type of post which can be either an image, video, or carousal album
datetime	Date & time of the Instagram post (extracted from timestamp)
date	Date of the Instagram post (extracted from timestamp)
time	Time of the Instagram post (extracted from timestamp)
shortened_caption	The first 20 characters of the actual Instagram post (extracted from caption)

Instagram Page Insights



The above screenshot shows the Instagram Page Insights webpage. The intent of this page is to communicate the performance of the organization's Instagram page overall.

Interactive elements & Visualization

The line chart on this page is used to communicate the Instagram page performance via various metrics over a period of time. The daterange filter adjusts the line chart to display metric values within the dates selected in the daterange filter. The new follower count tile shows the sum total of new followers of the Instagram page within the dates selected in the daterange filter.

Data

This webpage uses the "ZypInstagram_Insights1" (used by the line chart) and "ZypInstagram_Insights2" (used by the new follower count tile) Google Sheets documents that are saved on the Organization's Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

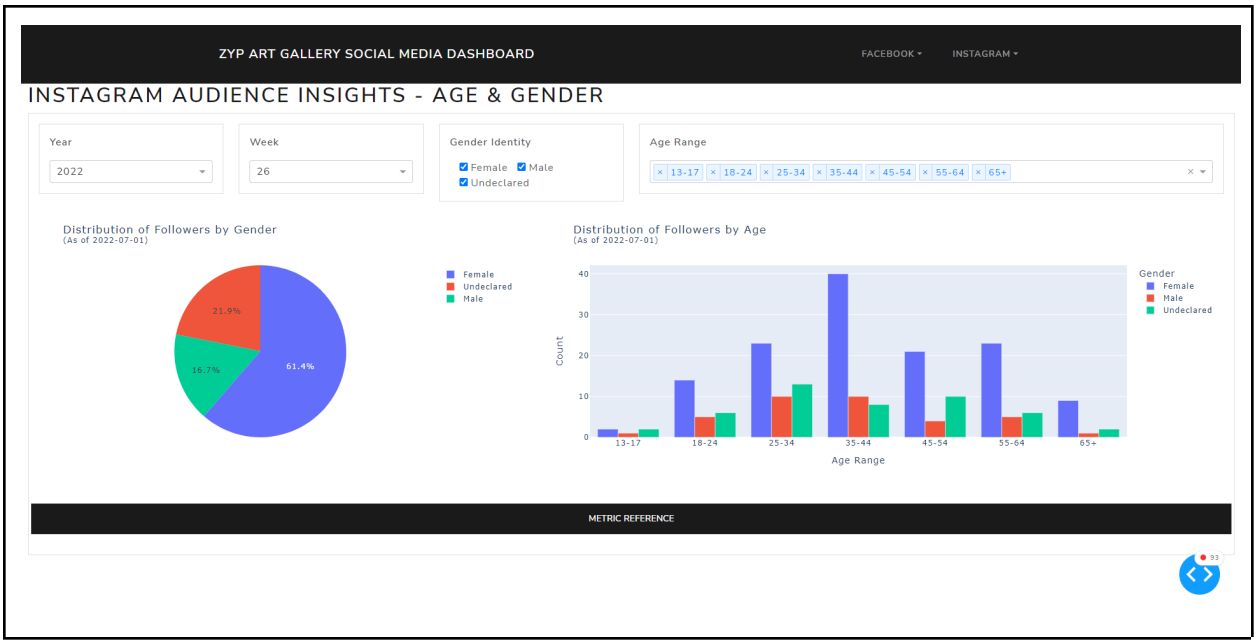
The following refers to information visualized on the new follower count tile.

Field	***Description***
end_time	Date
Metric	***Title & Description***
follower_count	*Follower Count* Total number of unique accounts following this profile

The following refers to information visualized on the line chart.

Field	***Description***
end_time	Date
Metric	***Title & Description***
impressions	*Impressions* Total number of times the Business Account's media objects have been viewed
reach	*Reach* Total number of times the Business Account's media objects have been uniquely viewed
email_contacts	*Email Contacts* Total number of taps on the email link in this profile
phone_call_clicks	*Phone Call Clicks* Total number of taps on the call link in this profile
text_message_clicks	*Text Message Clicks* Total number of taps on the text message link in this profile
get_directions_clicks	*Get Directions Clicks* Total number of taps on the directions link in this profile
website_clicks	*Website Clicks* Total number of taps on the website link in this profile
profile_views	*Profile Views* Total number of users who have viewed the Business Account's profile within the specified period

Instagram Audience Insights - Age & Gender



The above screenshot shows the Instagram Audience Insights - Age & Gender webpage. The overall intent of this webpage is to assess the performance of the Instagram page by age and gender.

Interactive elements & Visualization

As touched upon earlier in this section of the report, all audience related metrics are of period Lifetime per week. Thus, the daterange filter is replaced by the year & week dropdown filter. It functions in the same way as the daterange filter. The gender identity and age range filters function in a similar way as that of the corresponding Facebook section. Changing the selected option in the year and/or week dropdown filters would change the pie chart and side-by-side bar chart accordingly to reflect the metric value as of the selected year & week. The gender identity and age range filters would also further adjust the visualizations accordingly. By default the current year and prior to the current week is selected.

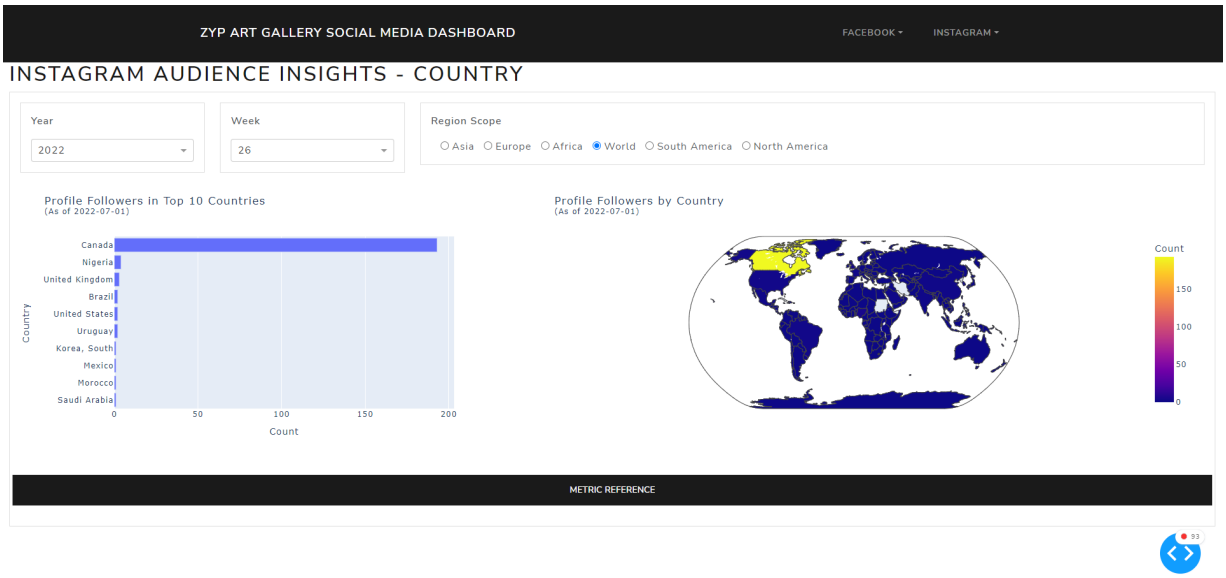
Data

This webpage uses the “ZypInstagram_Audience-Age&Gender” Google Sheets document that is saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
end_time	Date
year	The year the data was extracted (retrieved from end_time)
week	The week in the year the data was extracted (retrieved

	from end_time)
Metric	***Title & Description***
page_fans_gender_age [Gender abbreviation].[Age interval] E.g. F.13-17 (Female aged 13 to 17) • F = Female • M = Male • U = Unknown	*Gender and Age* The gender and age distribution of this profile's followers

Instagram Audience Insights - Country



The above screenshot shows the Instagram Audience Insights - Country webpage. The intent of this page is to communicate the performance of the organization's Instagram page by country.

Interactive elements & Visualization

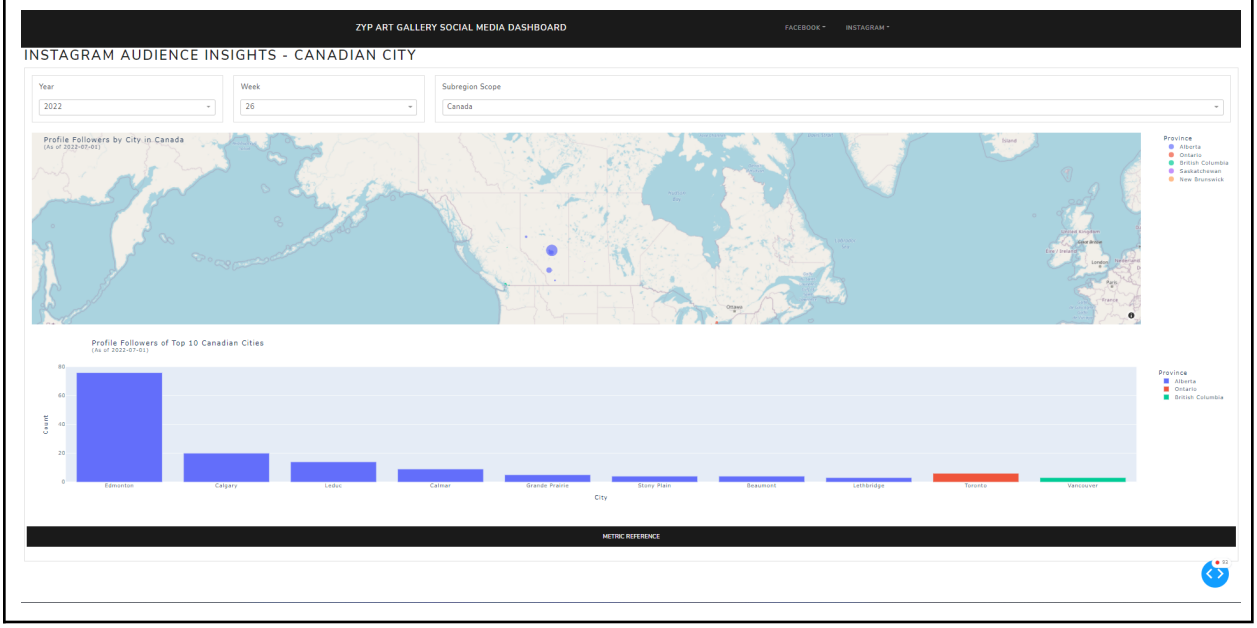
As touched upon earlier in this section of the report, all audience related metrics are of period Lifetime per week. Thus, the daterange filter is replaced by the year & week dropdown filter. It functions in the same way as the daterange filter. The region scope filter function in a similar way as that of the corresponding Facebook section. Changing the selected option in the year and/or week dropdown filters would change the choropleth and top 10 country bar chart accordingly to reflect the metric value as of the selected year & week. The region scope filter would also further adjust the visualizations accordingly. By default the current year and prior to the current week is selected.

Data

This webpage uses the “ZypInstagram_Audience-Country” Google Sheets document that is saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
end_time	Date
year	The year the data was extracted (retrieved from end_time)
week	The week in the year the data was extracted (retrieved from end_time)
Metric	***Title & Description***
audience_country Country name => E.g. South Africa	*Audience Country* The countries of this profile's followers

Instagram Audience Insights - Canadian City



The above screenshot shows the Instagram Audience Insights - Canadian City webpage. The intent of this page is to communicate the performance of the organization’s Instagram page by Canadian city.

Interactive elements & Visualization

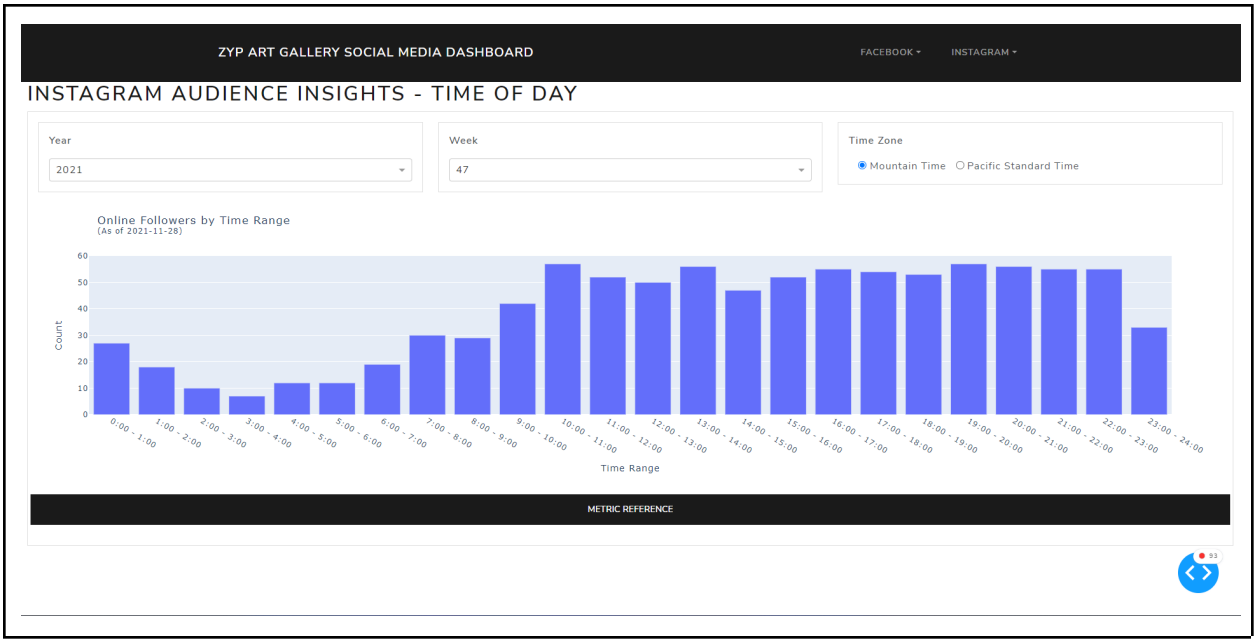
As touched upon earlier in this section of the report, all audience related metrics are of period Lifetime per week. Thus, the daterange filter is replaced by the year & week dropdown filter. It functions in the same way as the daterange filter. The subregion scope filter function in a similar way as that of the corresponding Facebook section. Changing the selected option in the year and/or week dropdown filters would change the bubble map chart and top 10 Canadian city bar chart accordingly to reflect the metric value as of the selected year & week. The subregion filter would also further adjust the visualizations accordingly. By default the current year and prior to the current week is selected.

Data

This webpage uses the “ZypInstagram_Audience-CanadianCity” Google Sheets document that is saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
end_time	Date
year	The year the data was extracted (retrieved from end_time)
week	The week in the year the data was extracted (retrieved from end_time)
Metric	***Title & Description***
audience_city City name, Province => E.g. Calgary, Alberta	*Audience City* The cities of this profile's followers

Instagram Audience Insights - Time of Day



The above screenshot shows the Instagram Audience Insights - Time of Day webpage. The intent of this page is to communicate the performance of the organization's Instagram page by time of day.

Interactive elements & Visualization

As touched upon earlier in this section of the report, all audience related metrics are of period Lifetime per week. Thus, the daterange filter is replaced by the year & week dropdown filter. It functions in the same way as the daterange filter. The time zone scope filter function in a similar way as that of the corresponding Facebook section. Changing the selected option in the year and/or week dropdown filters would change the bar chart accordingly to reflect the metric value as of the selected year & week. The time zone filter would also further adjust the visualizations accordingly. By default the current year and prior to the current week is selected.

Data

This webpage uses the "ZypInstagram_Audience-TimeOfDay" Google Sheets document that is saved on the Organization's Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
end_time	Date
year	The year the data was extracted (extracted from end_time)
week	The week in the year the data was extracted (extracted from end_time)

Metric	***Title & Description***
online_followers Time range in the 24 hour format => E.g. 14:00 - 15:00	*Audience City* The cities of this profile's followers

Project Codebase

Below is the GitHub repository of the custom dashboard application itself from my personal GitHub profile. The actual Google service account private key is not provided to protect the organization's data privacy. However, all of the Python scripts are intact and the codebase is visible to view. The current state of the repository and the commits made to it reflect the progress that was discussed in this section of the report.
https://github.com/kjeshang/ZypArtGallerySocialMediaDashboard

Pending Implementations

- Landing page
    - User interface/layout design
    - Potential content
    - Coding and internal testing
- Authentication
    - Further research
    - Coding and internal testing
- Deployment
    - Research
    - Practice and testing

Proposed Revisions

I am currently on track with my project as at the time of this writing, I have completed creating all of the Facebook and Instagram sections. There were minor changes requested on the Instagram Posts section's table. Specifically, it was requested that the shorted caption column values to be 40 to 50 characters instead of 20 characters. There was also some discussion with the analytics team leader regarding the overhead caused by the large data files used for the Canadian city sections of the dashboard application, but there is no definite solution as of yet. It was proposed to create a temporary dataframe/CSV upon selection of the daterange filter, then the temporary dataframe would be adjusted by the remaining filters to then eventually be used to create the visualizations.

There was more discussion regarding potential flow charts that would be helpful to understand how the dashboard applications function. The following link was provided by the analytics team leader. I will explore it more and hopefully incorporate in the next progress report.
https://developer.qualcomm.com/sites/default/files/docs/telematics/api/v1.46.10/a00005.html

I did encounter a Google Drive & Google Sheets API threshold error during development of the latter sections. It could be that each time running the application after stopping it, I was reaching the request limit to pull the social media data files, as well as related files from Google Drive. This made my application stop running due to the frequency of consequent stopping and starting it again. That being said, in its current finished form, the application runs fine. Potentially after the time frame of this course, I could explore making direct requests to Facebook Graph API retrieve data and create dataframes to be used for visualization, rather than relying on social media data files that are in Google Sheets or CSV file format. At this current time, I would like to move forward in researching authentication and deployment of the application at its current form. If time permits before the August 8th submission date, I will explore making direct requests to Facebook Graph API to retrieve data.

Midterm Progress Report

Proof of Progress

Development of the project is already underway as shown in the “Progress Report 1” section. As of 11th June 2022, I had intended to continue figuring out the Facebook Canadian City page of the application, but a lot of my time was taken up constructing documentation for this Midterm Progress Report section. However, I did have a meeting with the Zyp Art Gallery Analytics team on 9th June 2022. The Executive Director and Team leader were happy with my progress as the visualizations I constructed contained exactly what they were looking for. They appreciated how easy it was to utilize the interactive features to discover more insights over periods of time and as per category filtration. I did discuss the data possible data processing issue regading the Facebook Canadian City page of the application (discussed in Progress Report 1 section). They informed that if a bubble map chart is not possible, it would still be alright to simply have a bar chart that shows the top Canadian cities by the metric. They also recommended some of the charts and diagrams that will be seen in this section of the report. They also spoke of a potential idea whereby the data points could placed on the page metric line chart that would indicate the date and count of posts made. This would allow to track the time series of the page metrics in relation to the number of posts made at a given date.

Addressing Project Objectives

Below is a table that charts out whether I have fulfilled my objectives so far.

Objective	Explanation of module/features
Is the application easy to access by those that are part of the Zyp Art Gallery organization?	This objective is not fulfilled yet as detailed research and development has not started yet. I have conducted some brief research but it has been more exploratory at-best.

Is the application easy to use even for a non-technical person?	As my target group is primarily the Zyp Art Gallery Analytics team, to a certain degree they are all technical people but of all varied levels.
Is the application overall aesthetically pleasing?	The application is somwhat pleasing to the eye but I am unsure what my target group thinks as my conversation with them did not focus on aesthetics. I may have to work more on the color palette and theme.
Does the application follow a logical design in regards to navigation?	The application does follow a logical design. The project files are split across an index Python script that initializes the entire application (on a local machine) and renders the home page. There is an assets folder that has files that service to import datasets and hold reference data. There is an app folder that contains Python scripts that render the various webpages of the application.
Are the visualizations detailed & informative, yet easy on the eyes?	The visualizations created at this current time are detailed and informative enough according to the Zyp Art Gallery Analytics team members.
Do the visualizations have a good degree of interactivity in terms of date range, category, and/or any other identifier?	Yes it does. This proven by providing maximum filtration to adjust the date range via datepickers, metric category selections handled by checklists, radio buttons, and dropdowns. In turn, the visualizations respond to the changes made to the aforementioned interactive elements and adjust accordingly.

Environment Setup

Environment Setup to extract Social Media data

As aforementioned in earlier sections of this report, I have been volunteering at Zyp Art Gallery as of June 2021. Most of my work involved extracting Facebook and Instagram data. The basis of this project involves analyzing and visualizing the extracted social media data in a easy-to-use and accessible way. Thus, without having access to the social media data, the scope of this project may have been to wide to complete within a semester. The social media data itself was extracted using Python programming with the help of Facebook Graph API. In order to perform requests to Facebook Graph API, I needed to have a Facebook account and have a page role in the organization's Facebook page. I ended up making a Facebook account using my organization email. After creating a Facebook account, the executive director then assigned my Facebook profile to have a page role on the organization's web page. I was then

able to create a ‘Facebook for Developers’ account. I then had to create an ‘app’ on Facebook for Developers, and then state the purpose of it which was simply to make requests. To gain full access to information provided by various metrics, I had to provide Zyp Art Gallery’s privacy policy for approval before gaining full access.

Even though Facebook for Developers allows for making requests within the Graph Explorer, the drawback was that some request output was too long thus pagination would occur. In other words, the entire output would not be accessible within the explorer without further manipulation. This is further exemplified by the fact that the data output is in JSON format. Thus, making the API request with the help of Python programming was easier because I could code a loop that would paginate through the JSON output. In order to make requests to Graph API via a Python script, I needed to have an access token. Although, an access token is required overall to make requests to Graph API. Within Graph Explorer, an access token is provided by default but it is only valid for approximately two hours. Thus, using a short-lived access token is not ideal when running social media data extraction code on a weekly basis. And so, I extended the access token to last approximately two months. Unfortunately, Facebook for Developers do not provide lifelong access tokens. Thus, after the current token expires, I generate a new token and extend its lifetime. An access token with a longer lifetime was more ideal for me to create the social media data extraction code. This makes the act of running the social media data extraction code on a weekly basis easily executable on a weekly basis as I not have to keep on generating a new access token via the Graph Explorer. I only need to generate an access token with an extended lifetime every so often.

The organization utlilzes Google Drive as the primary office suite and data storage space at this current time. During the early days after I completed creating social media data extraction scripts for posts, page, & demographic insights, my final output would be a collection of data files in ‘CSV’ format. I used to manually upload them to a folder on the organization’s Google Drive. This used to be quite a tedious task. Thus, I decided to create a Google Service Account via the Google Cloud console. I then added the Google Drive and Google Sheets API. I then was able to acquire a private service account key which allows me to interact with Google Drive & Google Sheets. The next step was to create a dedicated Google Sheets file with a singular sheet that corresponds with the name of the data files in ‘CSV’ file format. The name of the Google Sheets file, sheet name, and identification value of the Google Sheets file (which is found in the URL), are utilized with the Google service account private key to take the contents of the data files in ‘CSV’ file format, and paste it into the respective Google Sheets files. The Python package called ‘gspread’ is imperative to auto-inserting the data to Google Sheets files.

As aforementioned, the social media data extraction code was created using Python. It was coded using VS Code. There are multiple scripts that are dedicated to extracting different types of metrics, yet they all follow a similar process with slight variations in terms of data cleaning & transformation. In practice, there is another Python script that runs all of the other extraction code scripts all at once. In essence, the Python scripts I wrote go through the following stages.

1. Setup

    a. Import the following packages: requests, pandas, time, datetime, dateutil, gspread.
    b. Create function to make request to Facebook Graph API.
    c. Create function that paginates through API request output if all output is not visible at once.
    d. Instantiate variables for API version and request URL.
    e. Retrieve current long-lived access token from text file.
2. Import data
    a. Retrieve social media data file which is in CSV format. The file consists of data that is until the last time the extraction code was run which is typically the week prior. It would take the form of a pandas dataframe.
    b. As there is a date column in the data file, the script takes the most recent date in the column (which is in the first row), and then instantiates as a variable to be used for the request made in the forthcoming stage.
    c. The recent date is then converted from datetime format into unix time format
3. Extraction
    a. Using the API request function along with the request URL & API version variables, and the variable holding the recent date, a request is made to Facebook Graph API to retrieve data of the specified metrics in the request. The output is instantiated to a variable and is of a JSON structure.
    b. Create column heading labels that would be used for the dataframe for the eventual dataframe
    c. Create function that is able to extract the date and metric values from a JSON object and appends to a list. There is a nested JSON object per date.
    d. Create function that applies the aforementioned function throughout the request output per nested JSON object in the form of a loop.
    e. Instantiate the aforementioned function to variable that would be used as the basis to create a dataframe of the newly extracted data.
4. Transformation
    a. The newly extracted data and column labels (created in the prior stage) are used to create a dataframe. The dataframe rows would be in descending order by date (i.e., the most latest date to earliest date).
    b. The dataframes of the imported data and newly extracted data are then combined together.
    c. Some data cleaning is performed to the combined dataframe such as dropping duplicates.
5. Loading
    a. The combined dataframe is then saved as a CSV file. In turn, overwriting the earlier imported social media data file.
    b. Then, the newly saved social media data file's contents are saved to the corresponding Google Sheets file on the organization's Google Drive. Specifically, the existing data in the Google Sheets file is deleted, and then the data in the CSV file are then copied and pasted in the Google Sheets file. This is accomplished through a function that takes parameters consisting of Google

Sheets file name, sheet name, and spreadsheet ID, along with authentication via the Google service account private key.

Below is a list of resources that were imperative to setting up an environment to retrieve social media data of the organization's Facebook and Instagram pages.

- Setting up Facebook for Developers and working with Graph Explorer
  - https://www.youtube.com/watch?v=LmhjVT9gIwk&list=PLhpgLgFy42uVqkUa_5P0HZIg0dwbVm96D&index=1
  - https://www.klipfolio.com/blog/facebook-graph-api-explorer
- Facebook Graph API access tokens
  - https://www.igorkromin.net/index.php/2020/10/02/facebook-access-token-error-validating-access-token-session-has-expired/
  - https://www.sociablekit.com/get-facebook-long-lived-user-access-token/
- Wrangling data from Facebook Graph API
  - https://bubbletao.com/2017/05/31/python-in-digital-analytics-automating-facebook-metric-reports/
- Metrics
  - https://supermetrics.com/api/getFields?ds=FB&fieldType=met
  - https://developers.facebook.com/docs/graph-api/reference/v11.0/insights#post_impressions
  - https://prezi.com/acd742tobfjg/quotnamequot-quotpage_fans_countryquot/
  - https://developers.facebook.com/docs/instagram-api/reference/ig-media/insights/
- Requests to Facebook Graph API and Python
  - https://stackoverflow.com/questions/37419788/difference-in-engaged-users-facebook-page-and-engaged-users-insights-api
  - https://stackoverflow.com/questions/38924707/get-post-insights-for-multiple-posts-in-one-call-using-graph-api-2-7/39104504
  - https://stackoverflow.com/questions/44491319/facebook-page-insights-api-to-retrieve-different-insight-metrics-using-python
  - https://stackoverflow.com/questions/38924707/get-post-insights-for-multiple-posts-in-one-call-using-graph-api-2-7/39104504
  - https://stackoverflow.com/questions/7227498/list-of-countries-and-cities-to-be-used-in-facebook-graph-api-for-targeting
- The full analysis process
  - https://medium.com/analytics-vidhya/analyse-your-personal-facebook-data-with-python-5d877e556692
  - https://www.youtube.com/playlist?list=PLhpgLgFy42uVqkUa_5P0HZIg0dwbVm96D
- Setting up Google service account via Google Cloud Console to use Python and Google Sheets
  - https://www.youtube.com/watch?v=cnPlKLEGR7E&list=LL&index=11
  - https://www.youtube.com/watch?v=bu5wXjz2KvU&list=LL&index=10
  - https://www.youtube.com/watch?v=TNIoGW8NzrY&list=LL&index=9
  - https://www.youtube.com/watch?v=2yIcNYzfzPw&list=LL&index=7

    - https://medium.com/craftsmenltd/from-csv-to-google-sheet-using-python-ef097cb014f9
- Specific ways to interact with Google Sheets using Python & gspread package
    - https://github.com/burnash/gspread
    - https://docs.gspread.org/en/latest/user-guide.html#updating-cells
    - https://github.com/PrettyPrinted/youtube_video_code/tree/master/2021/10/14/How%20to%20Use%20Google%20Sheets%20With%20Python%20(2021)
    - https://practicaldatascience.co.uk/data-science/how-to-read-google-sheets-data-in-pandas-with-gspread
    - https://readthedocs.org/projects/gspread-pandas/downloads/pdf/latest/

Environment Setup for Dashboard Application

The dashboard application's overall structure was described in the Progress Report 1 part within the Progress Implementation section of the report. In this section, the dependencies to ultimately create the dashboard application will be flushed out more concisely. Starting out, below is a list that contains all dependencies related to environment setup at this current stage of the project.

- Technology required
    - Computer with a Windows or Mac OS X operating system
- Knowledge required
    - Basic understanding of object oriented programming in regards to instantiating variables, creating & applying functions, creating classes
        - Knowledge & holistic understanding gained from the following Douglas College courses
            - CSIS 1175: Introduction to Programming I
            - CSIS 2175: Advanced Integrated Software Development
    - Understanding of software development in creating multi-layer applications
        - Knowledge & holistic understanding gained from the following Douglas College courses
            - CSIS 2300: Database I
            - CSIS 3300: Database II
            - CSIS 3175: Mobile Application Development I
            - CSIS 3275: Software Engineering
    - Basic understanding of descriptive statistics (E.g., calculating average)
        - Knowledge & holistic understanding gained from the following Douglas College courses
            - BUSN 2429: Business Statistics I
            - BUSN 3431: Business Statistics II
            - CSIS 3360: Fundamentals of Data Analytics

    - Data storytelling for the purpose of selecting an appropriate type of chart for a given dataset
        - Knowledge & holistic understanding gained from the following Douglas College courses
            - CSIS 3860: Data Visualization
    - Web development principles in terms of knowing various web elements and their respective html tags, as well as basic website design
        - Knowledge & holistic understanding gained from the following Douglas College courses
            - CSIS 1280: Multimedia Web Development
- Software
    - Python programming language version 3.8.9+; installed directly from the Python website or from the Anaconda version
        - Download link for latest Python distribution = https://www.python.org/
        - Download link for Anaconda distribution = https://www.anaconda.com/
        - Python scripts are used to code the project overall
        - Jupyter notebooks are used to demo certain portions of the dashboard application (e.g., data filtering, aggregation, visualization) that would eventually make its way into new/existing Python scripts to contribute to the creation
    - Microsoft VS Code; used as an integrated development environment
        - Download link = https://code.visualstudio.com/
    - Web browsers such as Google Chrome, Brave, or Firefox are used as the medium by which the dashboard application is functionable after running the code
        - Google Chrome download link = https://www.google.com/intl/en_ca/chrome/
        - Brave download link = https://brave.com/
        - Firefox download link = https://www.mozilla.org/en-CA/firefox/
- Additional dependencies (some of them were touched upon in more detail in the previous environment setup section)
    - Zyp Art Gallery Google account that allows access to the organization's Google Drive
    - Google service account private key for the purpose of authentication so that the dashboard application can import the social mediat data files directly from the organization's Google Drive
    - Country code ISOs that are in CSV file format (sources noted in References section of the report)
- Python packages (packages not pre-packaged by the Python are installed via Command Prompt/PowerShell/Terminal via Pip)
    - Gspread package; used to import the social media data files from the organization Google Drive
        - Installation = pip install gspread

- Pandas package; imported social media data files take the form of dataframes, and are also utilized as the basis of creating visualizations after performing any necessary aggregation and/or filteration
    - Installation = pip install pandas
- Numpy package; used to perform simple calculations such as finding the average of a given metric value from start date to end date selected
    - Installation = pip install numpy
- Datetime package: datetime, time, timedelta
    - Installation = pip install DateTime
    - Datetime used for converting string dates to appropriate datetime format; this is also helpful when handling datepickers
    - Timedelta is used to set the default start date to 30 days prior to the default end date
- Calendar package (imported but not actually used)
- Plotly package: express, graph objects
    - Installation = pip install plotly
    - Express used to create visualizations such as bar charts, line charts, pie charts, and area charts
    - Graph objects package used to create choropleths and other map related charts
- Dash package: Dash, dcc, html, Input, Output, State, Callback, Dash table
    - Instllation = pip install dash
    - Dash is used to instantiate the dashboard application so that is run using a local server via a web browser
    - Dcc is used to create web elements such as datepickers, dropdowns and checklists, as well as tabs
    - Html is used to create basic web elements such as headings and paragraphs
    - Input, Output, State, and Callback are used to reflect change on a webpage based on user manipulation of the interactive web elements such as those that serve the intent of filtration
    - Dash table is imported but not used
- Dash bootstrap elements package; used to create interactive elements such as dropdowns and buttons, as well as webpage structural elements such as containers, rows, columns, and card tiles
    - Installation = pip installation dash-bootstrap-components
- Sys package; used to import other Python script's variables and functions to be used in another Python script
    - Installation = pip install os-sys

Software Design Architecture

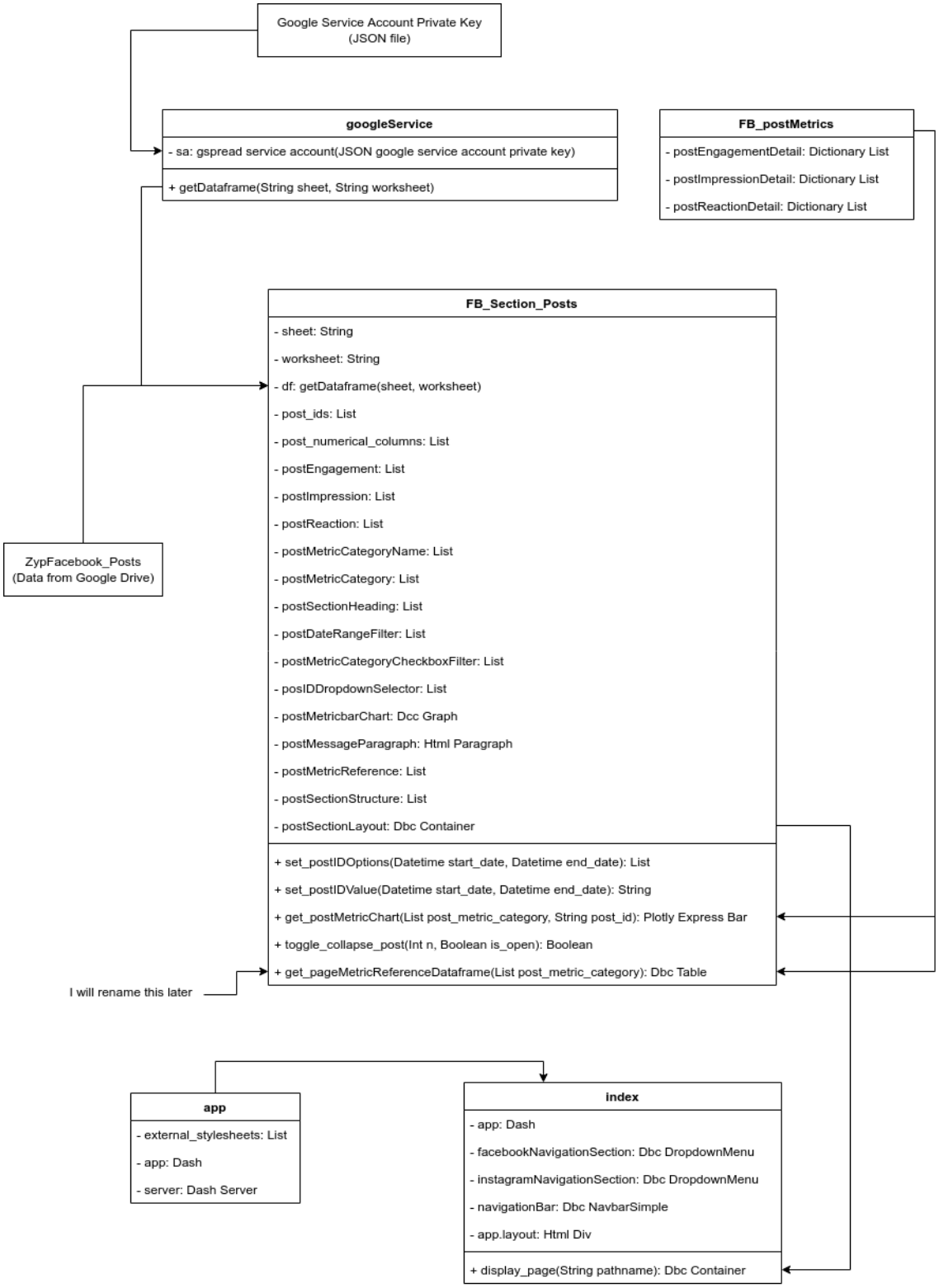
UML

Below is a list of diagrams that could best communicate the software design architecture of the project at its current state in an understandable way. Please note that the diagrams are subject to change upon final submission of the project.

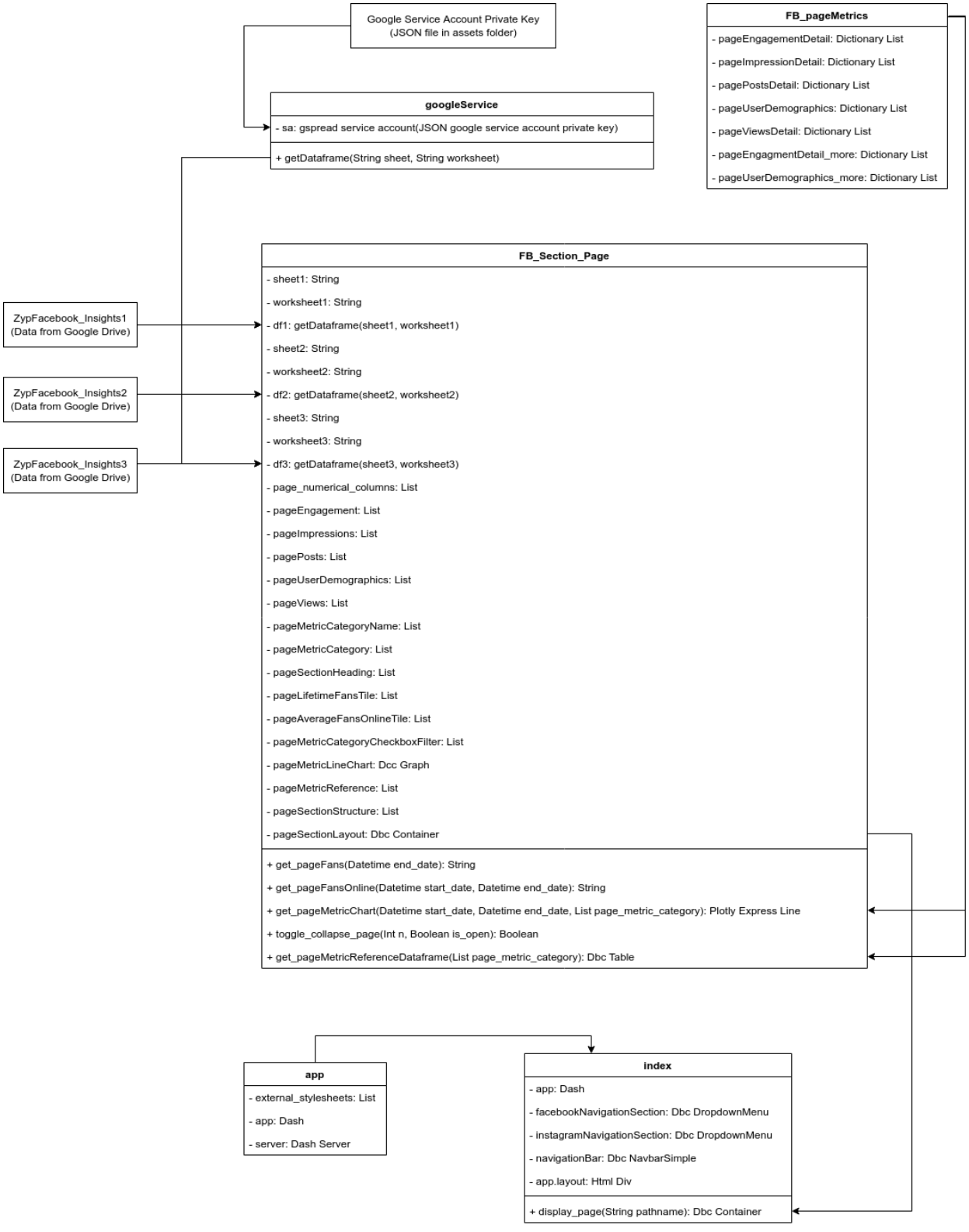
- Class Diagram
- Package Diagram
- Use Case Diagram
- Activity Diagram

Class Diagram

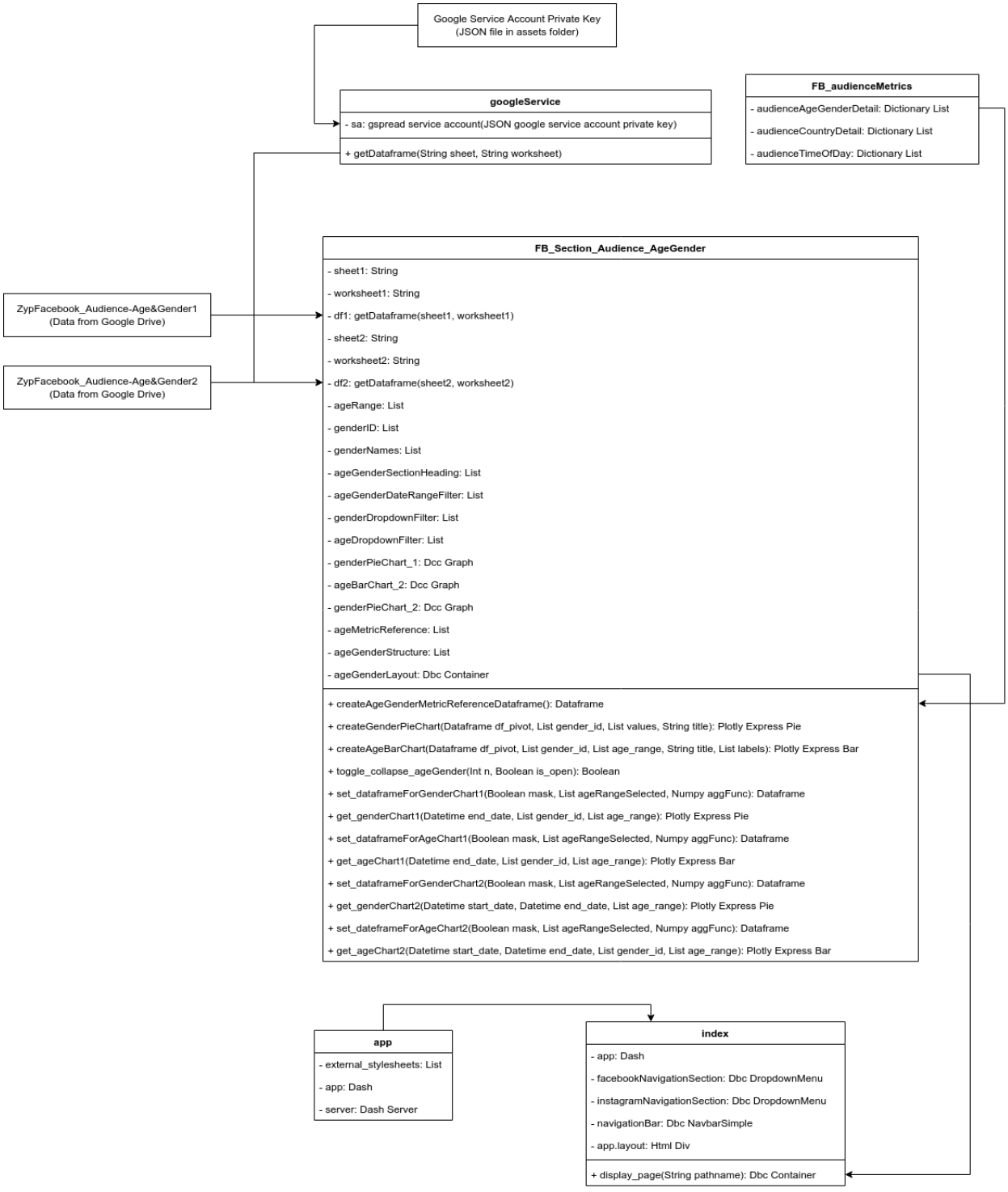
FB_Section_Posts



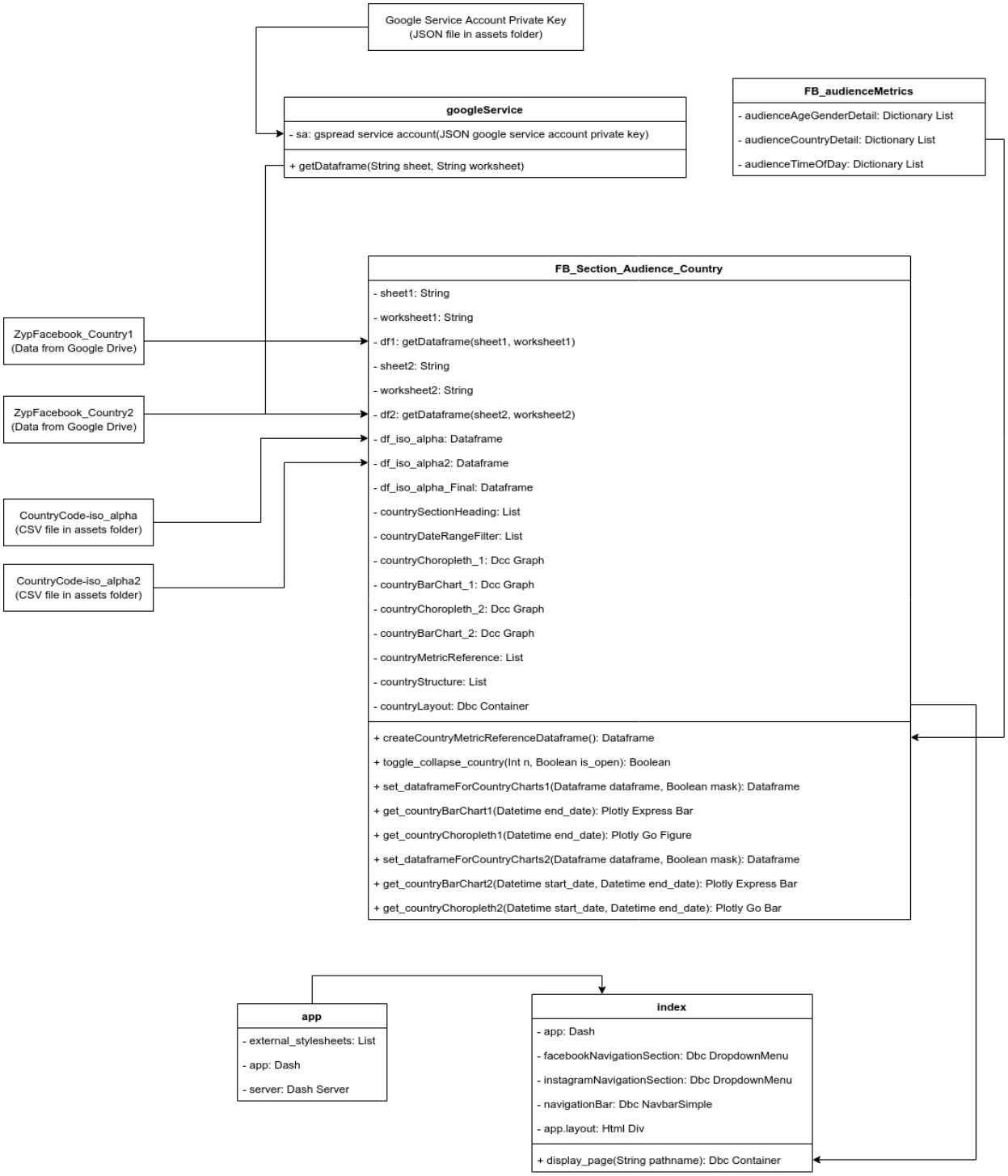
FB_Section_Page



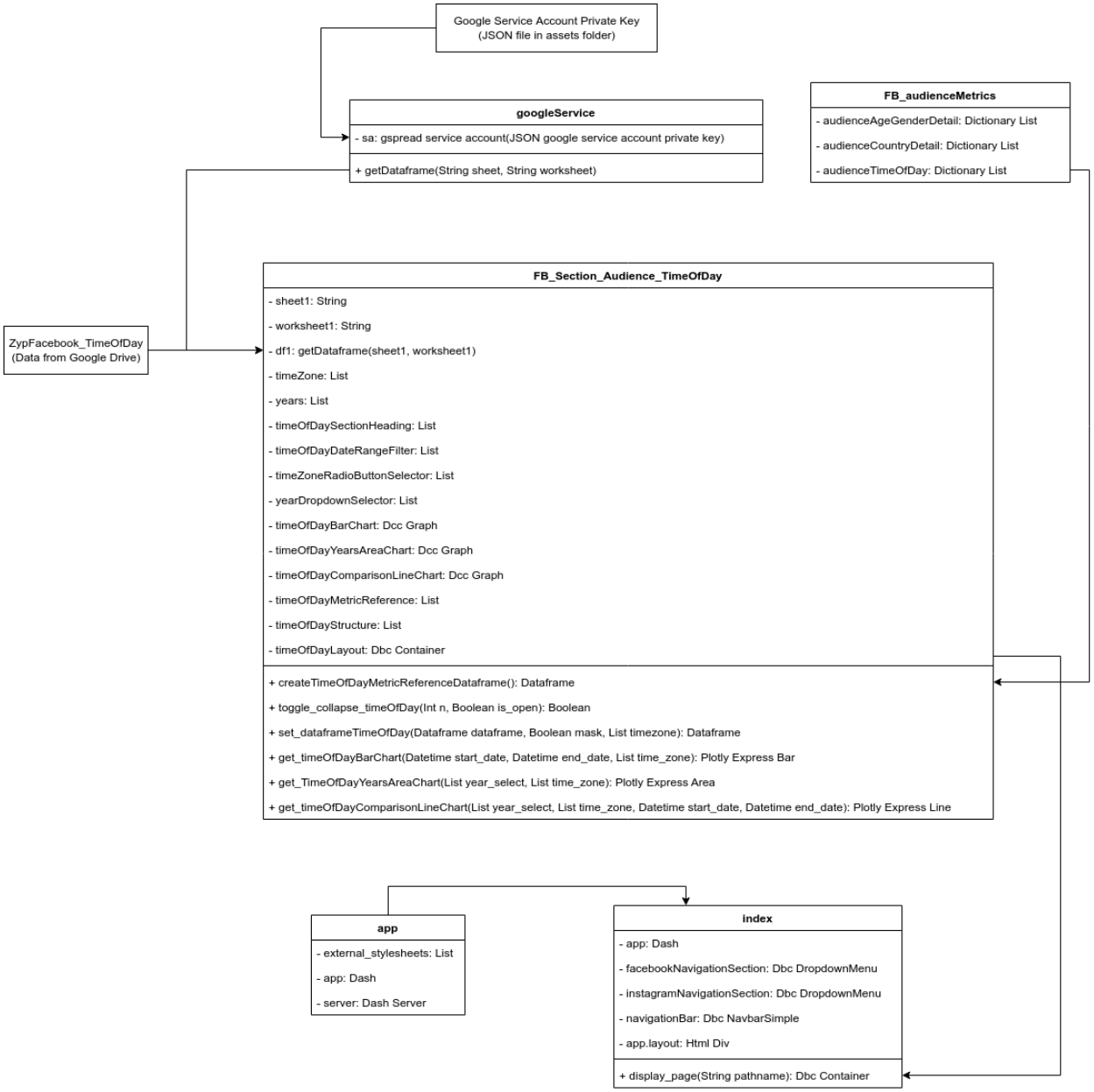
FB_Section_Audience_AgeGender



FB_Section_Audience_Country

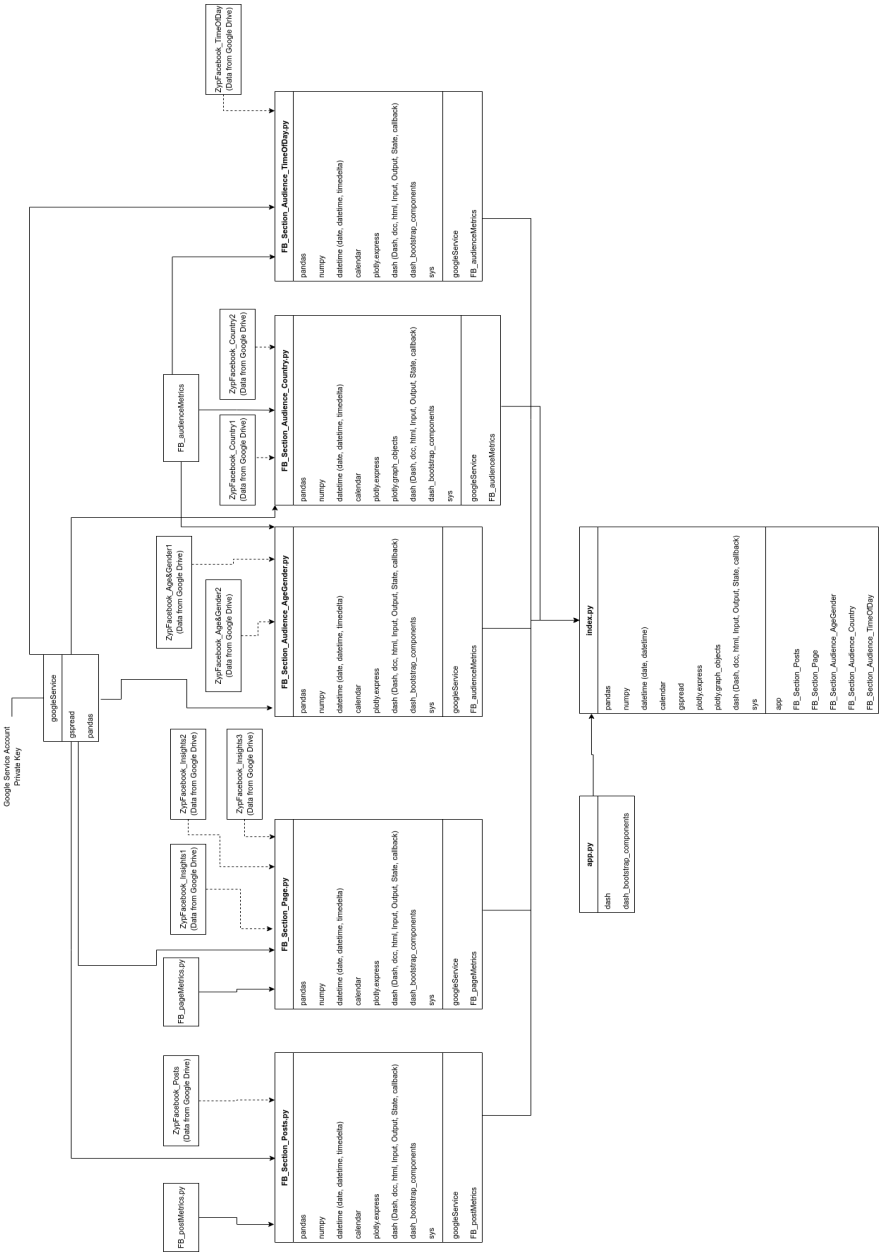


FB_Section_Audience_TimeOfDay

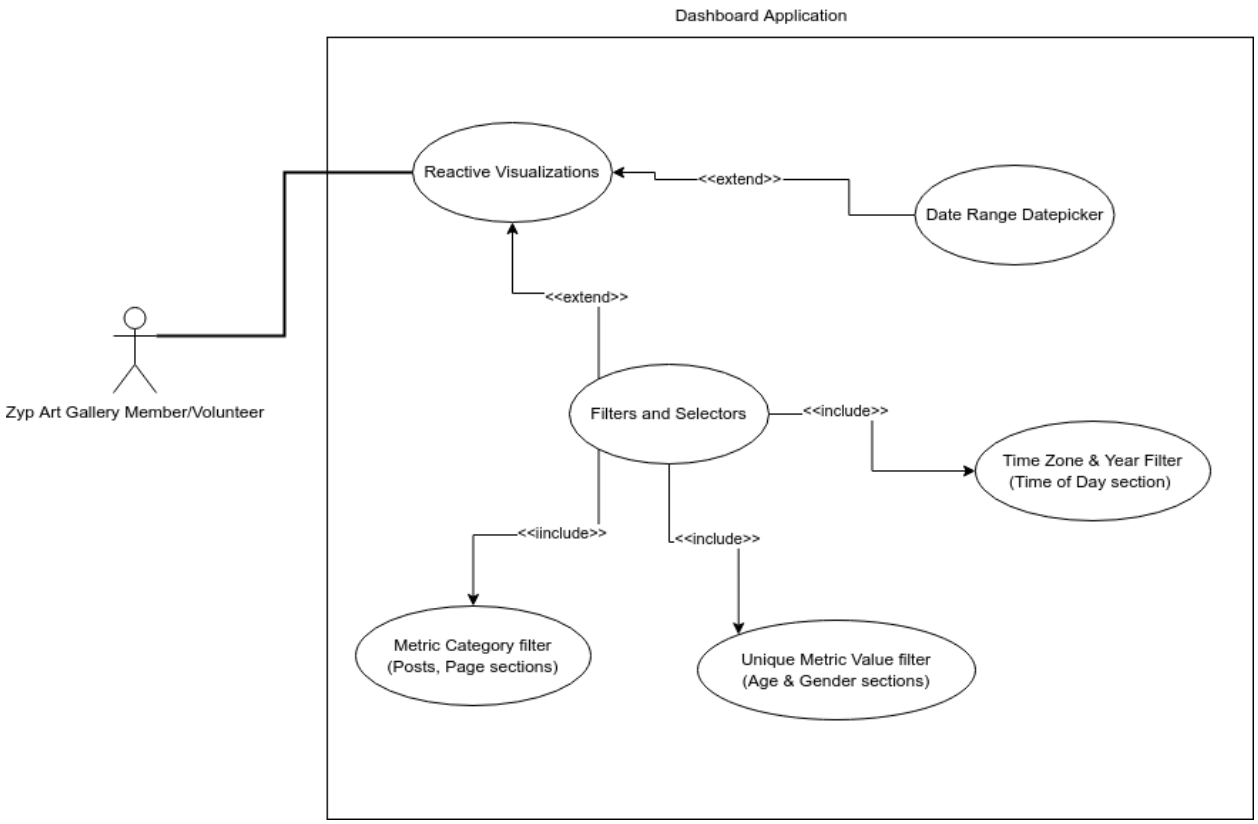


Package Diagram

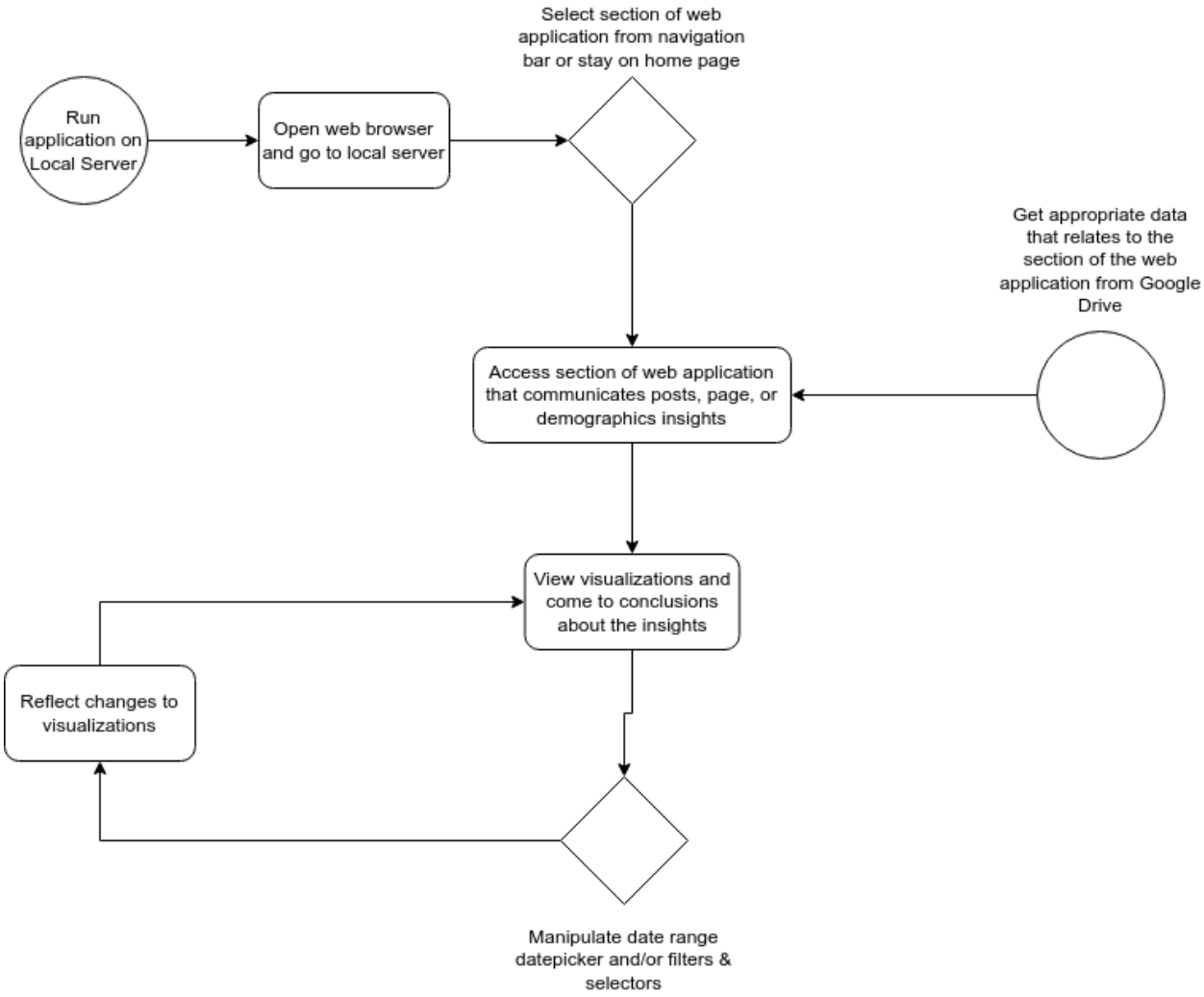
- Dashed line = Refers to a dependency on a data source
- Filled ‘regular’ line = Refers to a dependency from another script in the project



Use Case Diagram



Activity Diagram



Design Pattern

As explained in the earlier sections of the report, the main tool of use would be Dash, which is an “open-source library that facilitates the building of interactive web applications” using the Python programming language. The Dash library incorporates the Plotly package which is used

to visualize data in a way that is akin to Power BI and Tableau charts & diagrams. The great thing about Dash & Plotly is that you do not require any knowledge of JavaScript; only Python is needed which is easier to use and more user-friendly. Bootstrap, “which is an open-source CSS framework used for web development”, is compatible with Dash elements and is thus used to style the Dash app and make it aesthetically pleasing without the need of building an actual CSS stylesheet from scratch (Wan, 2020).

The scope of this project at this current time does not follow a specific software development design pattern in a traditional sense. However, Dash apps contain three key features that are imperative to constructing and running a web application successfully. Below are explanations for each of the three features.

1. Layout: This is a feature that tells a Dash app’s page how you want it to look like. This can include structural elements such as containers, rows, and columns, along with static web elements such as headings & paragraphs (in other words, lines of text), as well as web elements that are rendered with the intent of interactivity such as radio buttons, checklists, and dropdowns. Furthermore, the aforementioned elements can contain styling elements such as font, colors, and spacing.
2. Components: This is a feature that makes the app interactive. The layout feature may be in charge of instantiating and placing interactive web elements (e.g., radio buttons) on a webpage, but the component feature turns it interactive. For example, the radio buttons in the Facebook time of day webpage are there with the intent to change the visualizations based on the time zone selected from the radio button options (i.e., Mountain Time or Pacific Time).
3. Callbacks: This is a feature that takes the actions taken on the components (e.g., radio button), and performs a change to the webpage. In other words, it takes the the input of a component, and renders an output based on the specified input. In the example of the Facebook time of day webpage, the action taken on the time of zone radio button reflects the change on the connected visualizations.

Once all of the features are constructed and instantiated, the final step to turn the Dash app script to a running web application is to instantiate the app itself and run it on a local server on the computer. It is run on a local server for now as at this point in the project, I am still in the thick of the development stage (Wan, 2020).

As this project is a multi-page app, the dashboard application follows a structural pattern. There are designated scripts that are in charge of rendering the content of pre-defined webpage. For example, there is a specific script created to render the Facebook posts section of the web application, and there is another specific script created to render the Facebook page section of the web application. Each script created to render a section of the web application are in a single folder called “apps”. In addition, there is also an “assets” folder that contains miscellaneous Python scripts that are imperative to the scripts in the “apps” folder, as well as other important files. The contents of the “assets” folder currently consist of the following.

- Google service account private key as a JSON file.

- Google service Python script that takes the Google service account private key and creates a function with the intent to retrieve a specified social media data file from Google Drive.
- Facebook metrics Python scripts for post, page, and demographic that contain various dictionary lists of sub-categories containing metric names, titles, and description. These scripts are helpful to create the metric reference table and hover information on visualizations.
- Country code ISOs that is used in the data aggregation and preparation code when rendering the Facebook country section of the web application. ISO codes make the choropleth to actually make the countries visible and show distinction.

As explained, the scripts in the “apps” folder render the actual webpages in the web application. All of these scripts follow a similar sequential structure as shown below.

1. Import dependencies
    a. Import Python packages
    b. Import any necessary variables from other Python scripts in the assets folder
2. Import the data
    a. Import the required social media data file/s from Google Drive as a pandas dataframe
3. Prepare the data
    a. Convert the date related column of imported data from string type to datetime type if necessary
    b. Segment columns of imported data into categories if necessary to accommodate for potential-future filtration
    c. Create option labels to be used for interactive web elements that are related to metric category filtration (if needed)
4. Create elements of the webpage
    a. Variables used to instantiate both static and interactive web elements
    b. Static web elements have fully defined parameters
    c. Interactive web elements such as filtration components and/or visualizations have full or partial parameters along with identification elements used that are used by the callbacks to return a specified output
    d. Most of the web elements instantiated are within a Python list so as to group together related elements together to make it easier to place in the variable that instantiates the structure of the page
5. Page layout
    a. Create the skeleton/structure of the webpage that also puts in the variables that instantiate the elements of the webpage
    b. Instantiate the webpage structure to a variable dedicated to layout
6. Callbacks
    a. Contain various functions designed to take the input of various components and return a specified output

Finally, there is a Python script in the main directory of the project. This script is dedicated to running the entire web application on a local server. Thus, it imports all of the layout related variables from the scripts in the “apps” folder. This script also renders a navigation bar. Each link

in the navigation bar will correspond to a script in the “apps” folder. Upon selection of a specific link, the corresponding script’s layout would be rendered in the web application, thus, functioning as a multi-page app. The main directory also contains an “app.py” and an “_init_.py” file. Currently, there is not much use of it in the development stage, but they may be important down the road when it comes time to attempt deployment of the dashboard application based on my research (Wan, 2020).

Scripts

The link below is the GitHub repository of the Zyp Art Gallery social media extraction code. It is on my public GitHub profile. There is fully functioning version of my extraction code on the organization’s GitHub profile but it is private. The version on my GitHub profile does not contain the actual access token or Google service account private key. This is to protect the data privacy of the organization.
https://github.com/kjeshang/ZypArtGallery-SocialMediaDataExtraction

Below is the GitHub repository of the custom dashboard application itself from my personal GitHub profile. The actual Google service account private key is not provided to protect the organization’s data privacy. However, all of the Python scripts are intact and the codebase is visible to view.
https://github.com/kjeshang/ZypArtGallerySocialMediaDashboard

Video Presentation

Below is the link to the midterm video presentation. If it is not accessible, please kindly reach out to me.
https://collegedouglas-my.sharepoint.com/:v:/g/personal/jeshangk_student_douglascollege_ca/EQwTbT-aFahBuCQE2TIlpWcBIIU5lPQ33efPZrj5kp-QaQ?e=bh1gtS

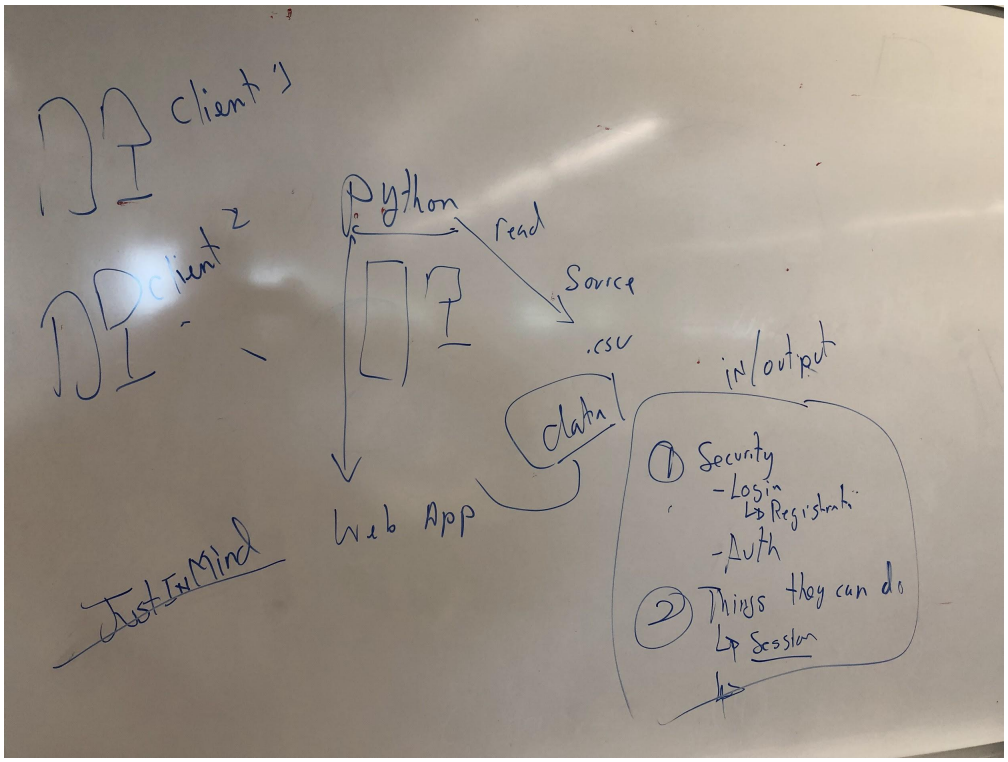
Progress Report 1

After receiving approval for the project proposal, this period consisted mainly of constructing the design of the Facebook section’s user interfaces and exploring potential visualizations. I used Draw.io to draft webpage layouts. After constructing baseline layouts, I began performing technical research and proceeded with coding the various webpages related to the Facebook section. I managed to complete coding Facebook section’s webpages that involve Facebook post insights and page insights, as well as audience insights related to age & gender, country, and time of day. Prototyping of filteration, aggregation, and visualization code was done in a Jupyter notebook before creating a dedicated Python script file that is part of the application’s codebase structure. Internal testing and layout design adjustments were performed whilst simultaneously coding. In regards to internal testing, if there is a failure of some sort, a visualization would fail to render or a visualization would not change based on the user manipulation/change of an interactive element. A failure would result in a Dash callback error; except for “ID in layout not found”. I made a start in researching and coding Facebook audience insights webpage for Canadian city. Within this period of time, I spoke to my instructor for

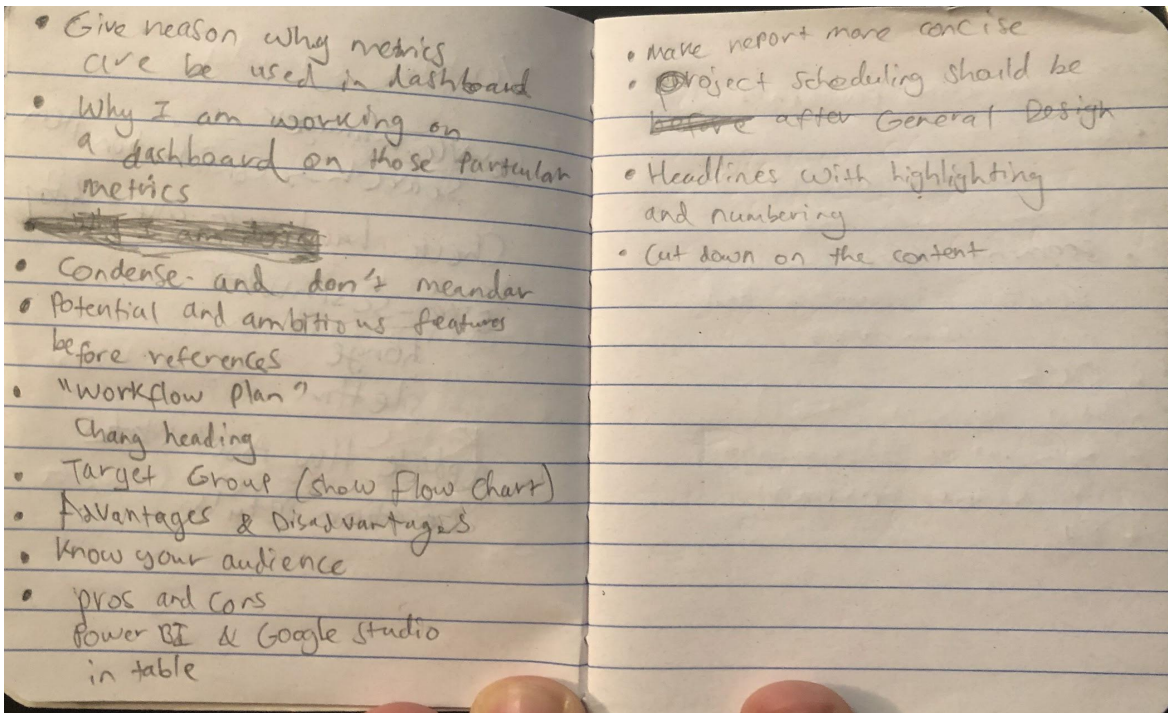
general advice on drafting web layouts and project codebase structure. He explained his thoughts to me verbally with the aid of writing on the whiteboard so that I could visually understand. Also, I had one virtual meeting with the Zyp Art Gallery Analytics team post-submitting the project proposal. Although, discussion was more about my project proposal structure rather than the project itself. I unfortunately have not gotten a chance to have more virtual meetings due to lack of availbility of the other Analytics volunteers. The sections below show evidence of the progress made for the project between May 20th and June 6th.

Evidence

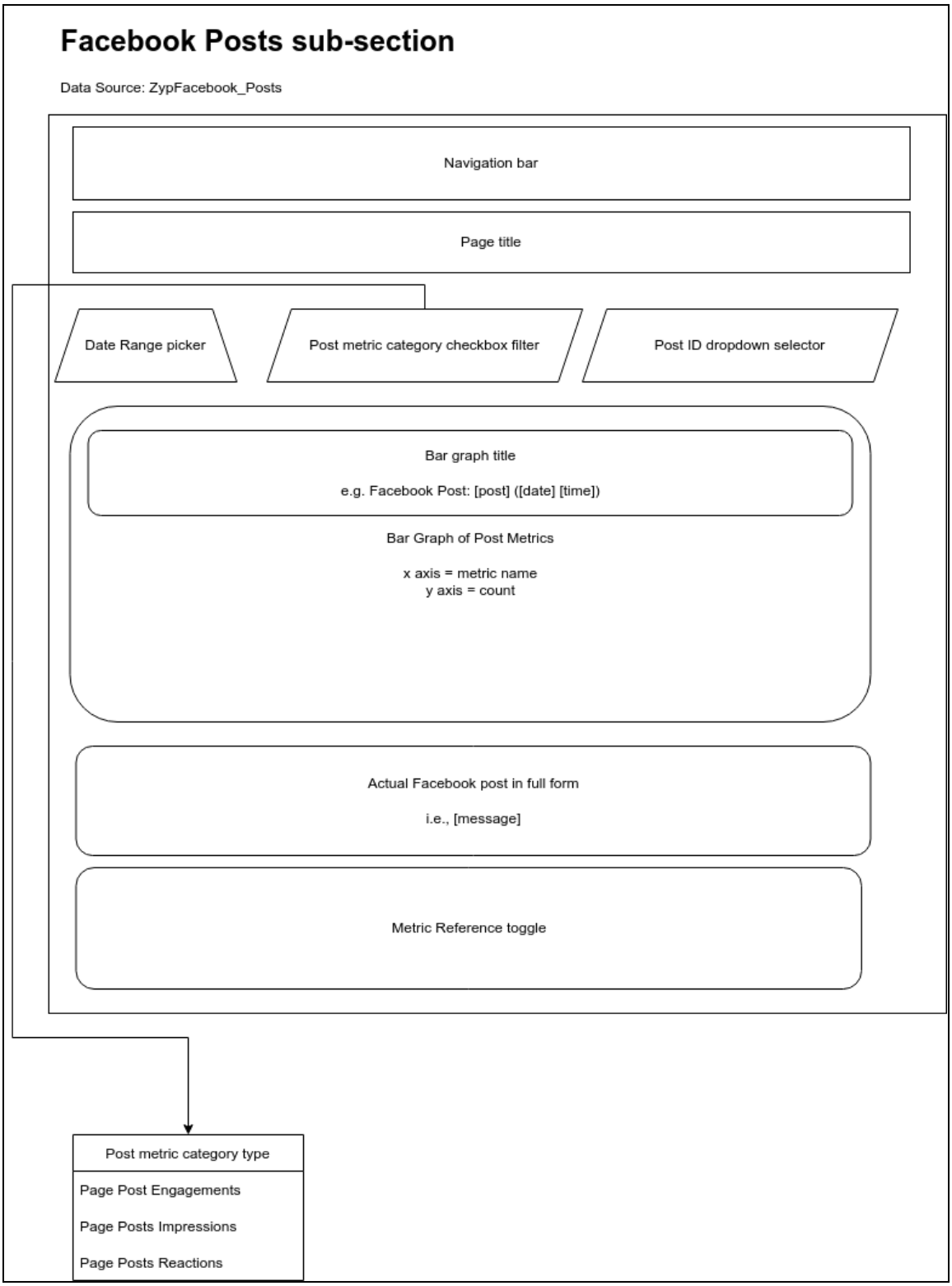
Discussion with Instructor



Notes from Zyp Art Gallery Analytics meeting



Facebook section webpage User Interface designs

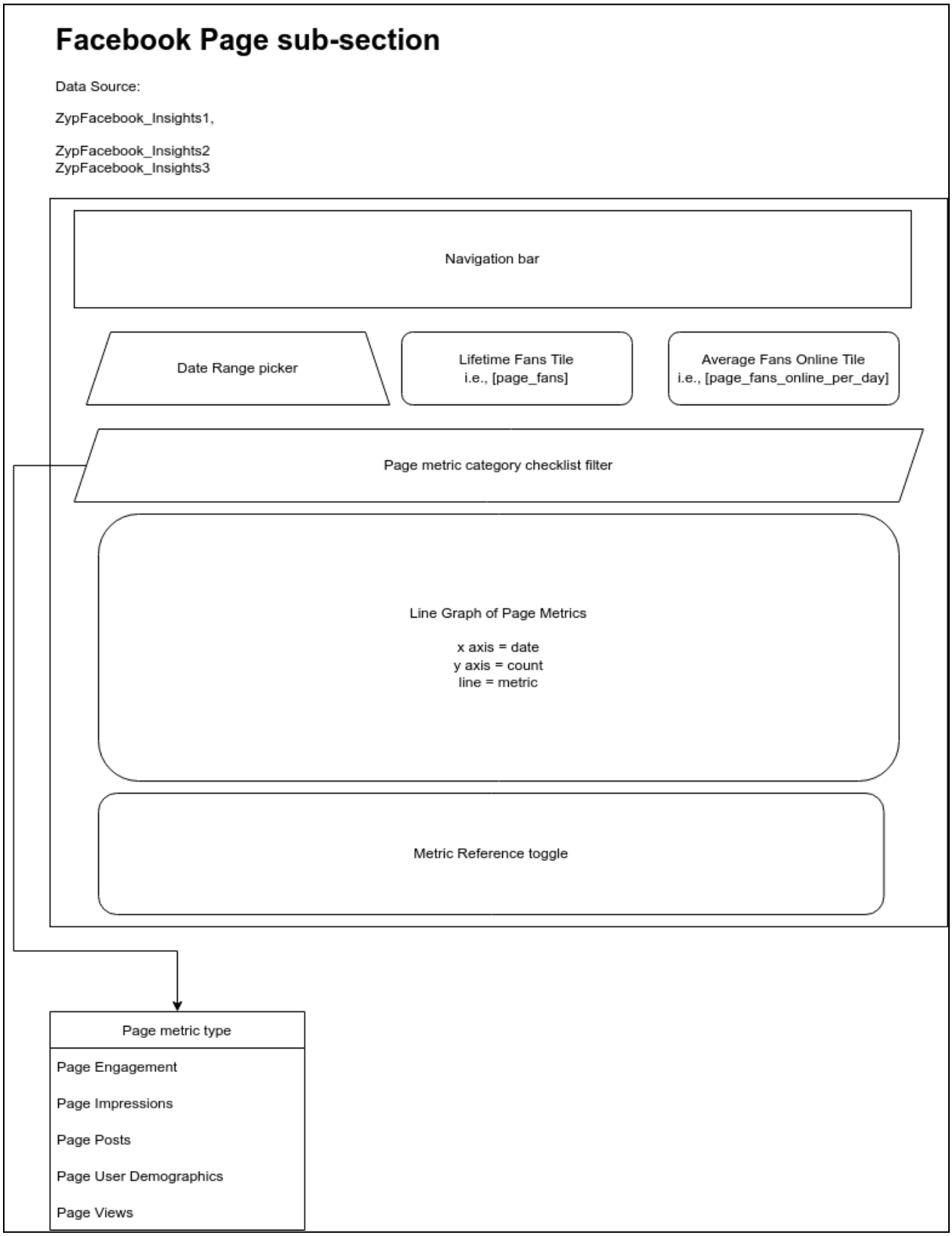


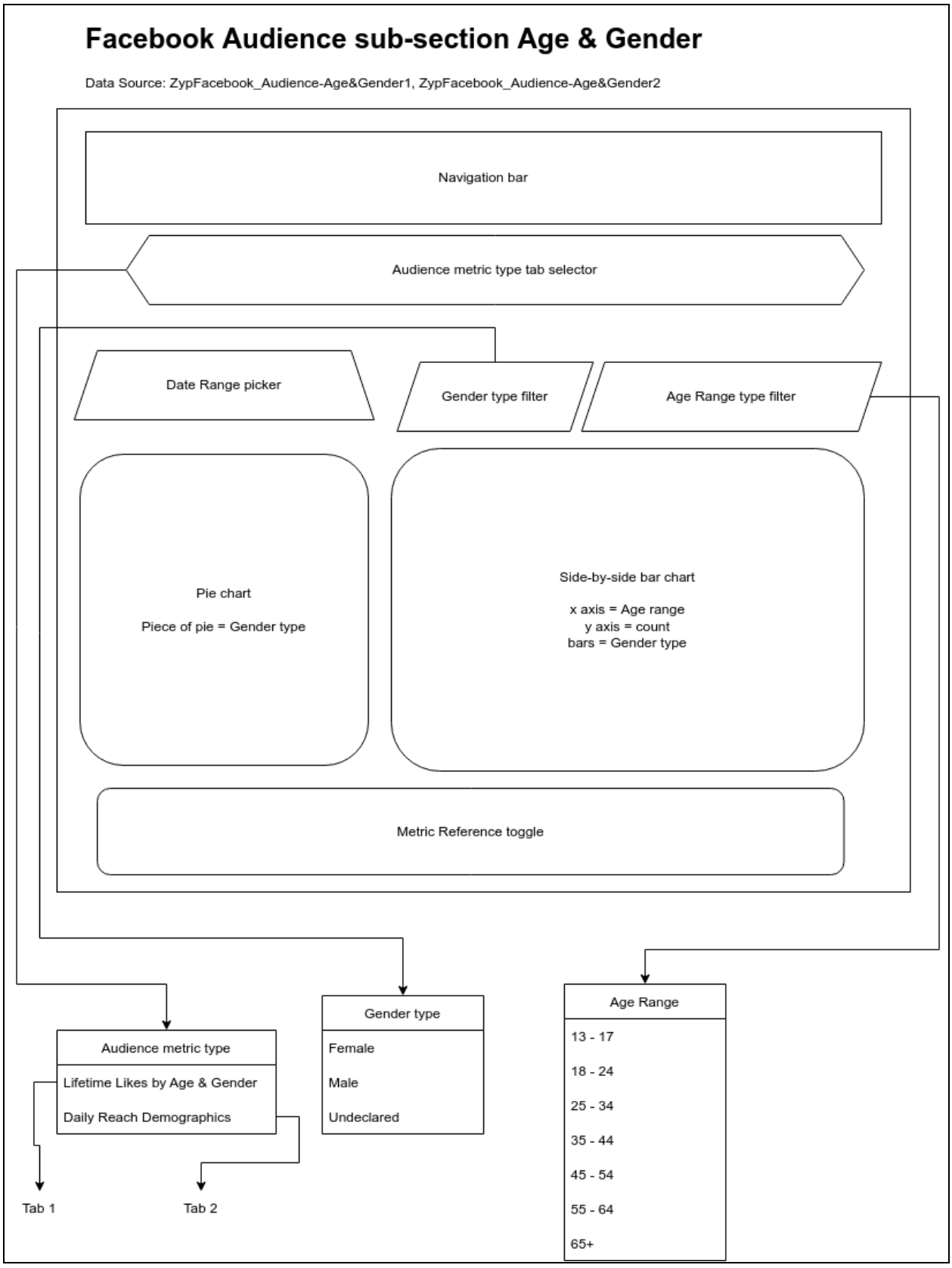
Facebook Page sub-section

Data Source:

ZypFacebook_Insights1,

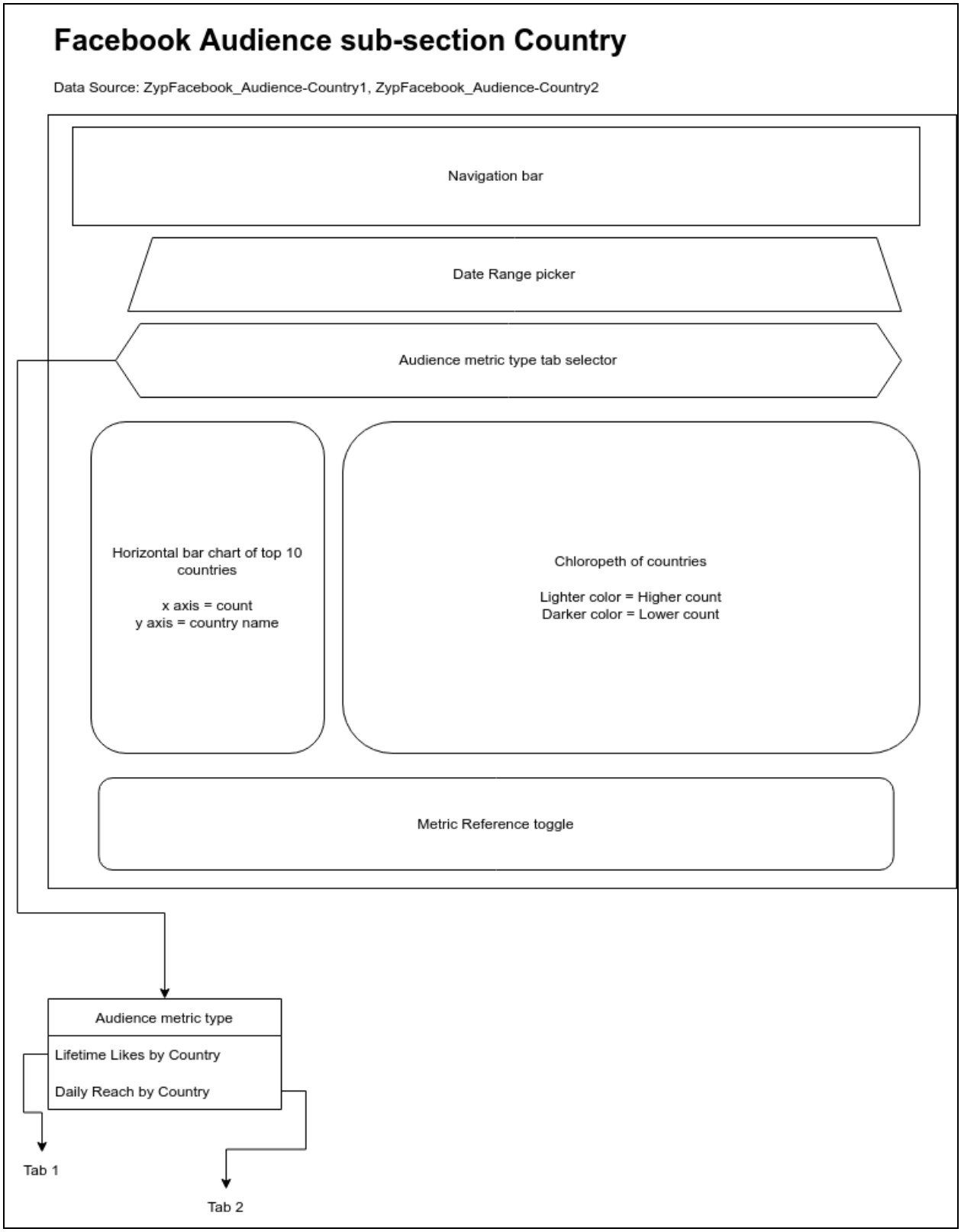
ZypFacebook_Insights2
ZypFacebook_Insights3




Facebook Audience sub-section Age & Gender
Data Source: ZypFacebook_Audience-Age&Gender1, ZypFacebook_Audience-Age&Gender2
Navigation bar
Audience metric type tab selector
Date Range picker
Gender type filter
Age Range type filter
Pie chart
Piece of pie = Gender type
Side-by-side bar chart
x axis = Age range
y axis = count
bars = Gender type
Metric Reference toggle
Audience metric type
Lifetime Likes by Age & Gender
Daily Reach Demographics
Tab 1
Tab 2
Gender type
Female
Male
Undeclared
Age Range
13 - 17
18 - 24
25 - 34
35 - 44
45 - 54
55 - 64
65+

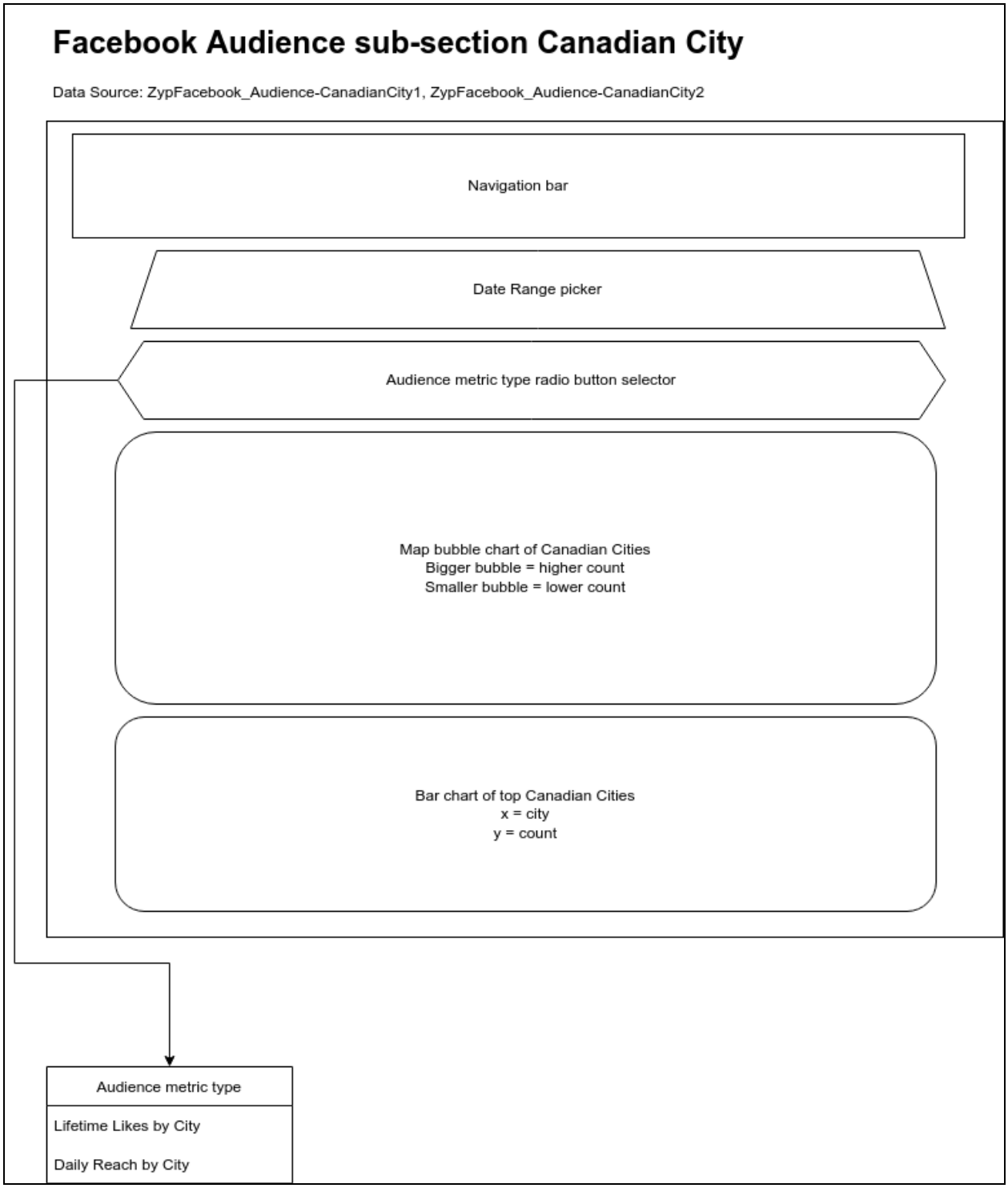
Facebook Audience sub-section Country

Data Source: ZypFacebook_Audience-Country1, ZypFacebook_Audience-Country2



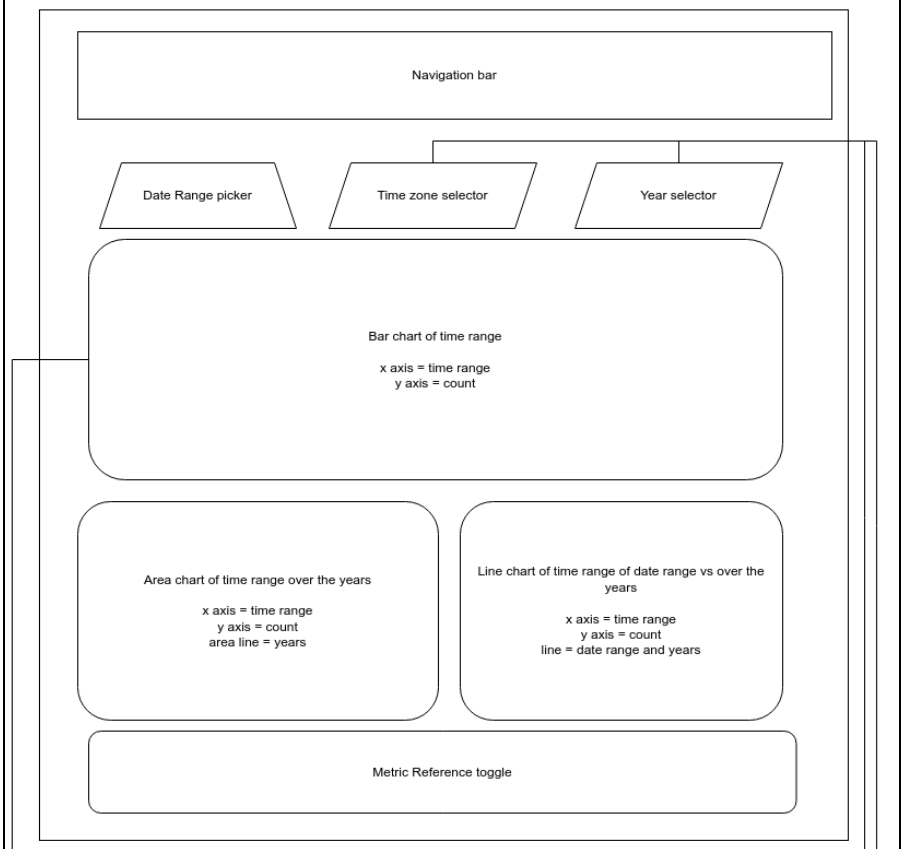
Facebook Audience sub-section Canadian City

Data Source: ZypFacebook_Audience-CanadianCity1, ZypFacebook_Audience-CanadianCity2



Facebook Audience sub-section Time of Day

Data Source: ZypFacebook_Audience-TimeOfDay



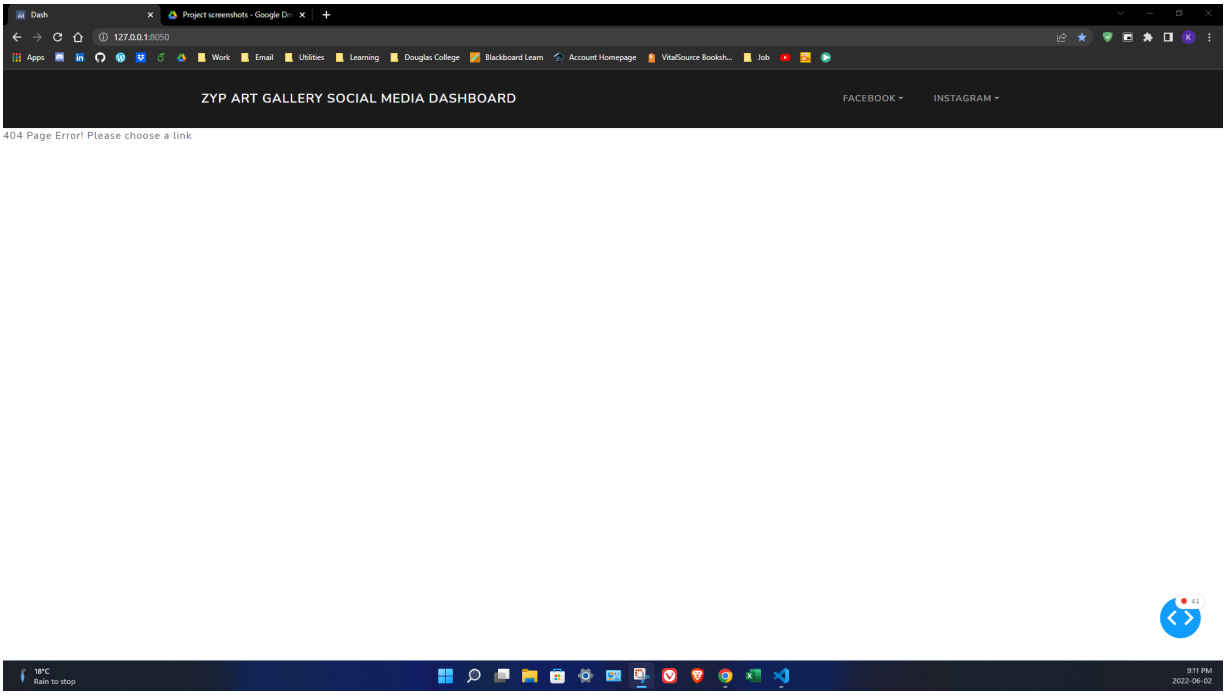
Time Range
0:00 - 1:00
1:00 - 2:00
2:00 - 3:00
3:00 - 4:00
4:00 - 5:00
6:00 - 7:00
7:00 - 8:00
8:00 - 9:00
9:00 - 10:00
10:00 - 11:00
11:00 - 12:00
13:00 - 14:00
14:00 - 15:00
15:00 - 16:00
16:00 - 17:00
17:00 - 18:00
18:00 - 19:00
19:00 - 20:00
20:00 - 21:00
21:00 - 22:00
22:00 - 23:00
23:00 - 24:00

Year
Current year (e.g. 2022)
Current year - 1 (e.g. 2021)
Current year - 2 (e.g. 2020)
...
2018

Time Zone
Mountain Time
Pacific Standard Time

Application screenshots

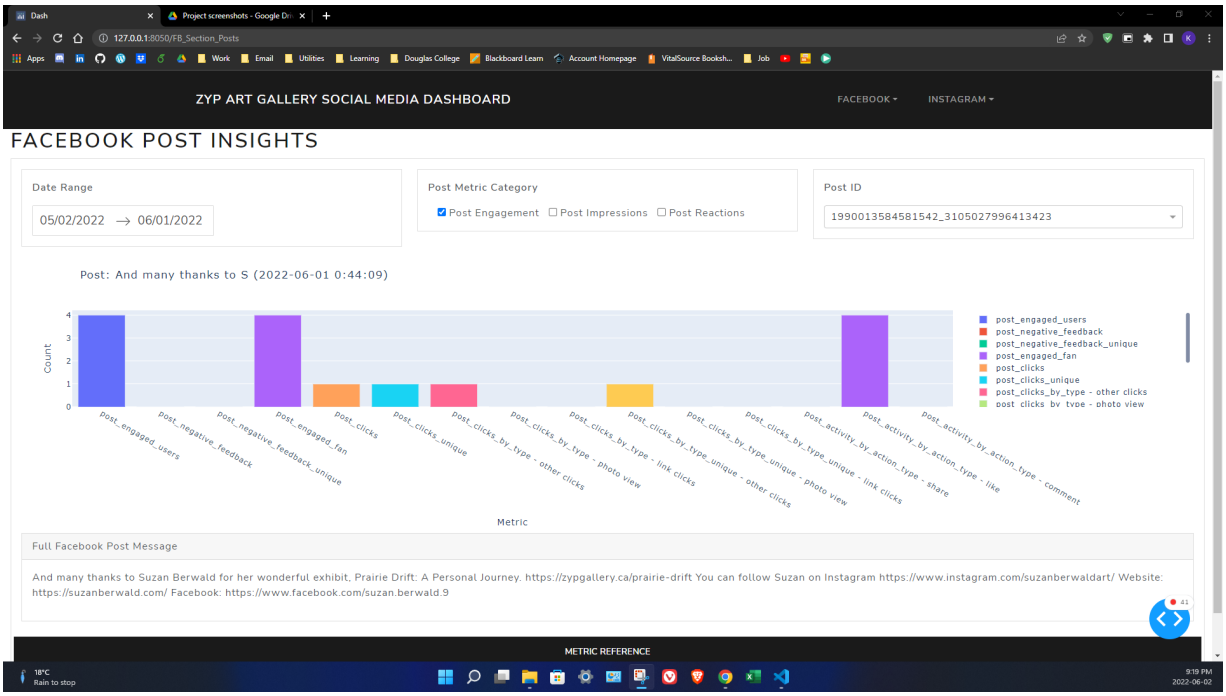
Home page (i.e., index) (to be completed on a later date)



As can be seen above, the homepage content is yet to be decided. For now, whenever the application is started and the home page is accessed, the following sentence appears: “404 Page Error! Please choose a link”. In addition notice that the navigation bar appears at the top of the webpage. This navigation bar is standard throughout the rest of of the webpages in subsequent Facebook & Instagram sections. Although the Instagram section’s not being designed or coded yet. However, the navigation bar is consistent for all of the webpages present in the Facebook section. As of the time of this writing, the following is the navigation bar link hierarchy.

- Zyp Art Gallery Social Media Dashboard application (title hyperlinked to the home page)
- Facebook section navigation dropdown menu
    - Posts
    - Page
    - Audience (this is not a link but a header within navigation dropdown)
        - Age & Gender
        - Country
        - Canadian City (no link but webpage in progress of development; currently reverts to home page)
        - Time of Day
- Instagram section navigation dropdown menu (**TBD**)

Facebook Post Insights



As can be seen in the above screenshot shows the Facebook Post Insights webpage. The intent of this page is to communicate the performance of the organization's Facebook posts.

<u>Interactive elements</u>

The webpage contains two interactive elements that are consistent throughout the other webpages as well: the Date Range datepicker and the Metric Reference button. Metric Reference button toggles a table that appears below it explaining the meaning of the selected metrics that are visible on the visualizations in terms of title and description. The datepicker on this webpage filters the options available in the Post ID dropdown. Specifically, the Post ID refers to a unique identifier for the Facebook posts on the organization's Facebook page. Thus, the Post ID dropdown contains unique identifiers of Facebook posts that were published within the selected date range from the datepicker. Selecting a specific Post ID from the respective dropdown would change the bar chart visualization to display the metric value counts of the selected Post ID. The title of the bar chart visualization would also change to display the first 20 character's of the Post ID's message (i.e., actual Facebook post) along with its respective date & time of post. The box showing the full Facebook post, which is below the bar chart, would change to show the entire post message of the selected Post ID. Also notice the Post Metric Category checklist. This checklist groups the various post metrics together based on engagement, impressions, and reactions. The metrics associated with the three categories help to assess the engagement (i.e., overall interactivity with the post), impressions (i.e., reach), and reactions (e.g., like) of the selected post. Selecting all or non of the options in the Post Metric Category checklist would also change the metrics visible in the table that is toggled by the Metric Reference button.

Data

This webpage uses the “ZypFacebook_Posts” Google Sheets document that is saved the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
id	Identification number of the facebook
message	The actual post made to facebook
created_time	Date & time of facebook post
date	Date of facebook post (extracted from created_time)
time	Time of facebook post (extracted from created_time)
post	The first 20 characters of the actual facebook post (extracted from message)

Post Engagement:

Metric	***Title & Description***
post_engaged_users	*Lifetime Engaged Users* Lifetime: The number of unique people who engaged in certain ways with your Page post, for example by commenting on, liking, sharing, or clicking upon particular elements of the post. (Unique Users)
post_negative_feedback	*Lifetime Negative Feedback* Lifetime: The number of times people have given negative feedback to your post. (Total Count)
post_negative_feedback_unique	*Lifetime Negative Feedback from Users* Lifetime: The number of people who have given negative feedback to your post. (Unique Users)
post_engaged_fan	*Lifetime People who have liked your Page and engaged with your post* Lifetime: The number of people who have liked your Page and clicked anywhere in your posts. (Unique Users)

post_clicks	*Lifetime Matched Audience Targeting Consumptions on Post* Lifetime: The number of clicks anywhere in your post on News Feed from the user that matched the audience targeting on it. (Total Count)
post_clicks_unique	*Lifetime Matched Audience Targeting Consumers on Post* Lifetime: The number of people who matched the audience targeting that clicked anywhere in your post on News Feed. (Unique Users)
post_clicks_by_type - other clicks post_clicks_by_type - photo view post_clicks_by_type - link clicks	*Lifetime Matched Audience Targeting Consumptions by Type* Consumption Type: other clicks, photo view, link clicks Lifetime: The number of clicks anywhere in the post on News Feed from users that matched the audience targeting on the post, by type. (Total Count)
post_clicks_by_type_unique - other clicks post_clicks_by_type_unique - photo view post_clicks_by_type_unique - link clicks	*Lifetime Post Audience Targeting Unique Consumptions by Type* Consumption Type: other clicks, photo view, link clicks Lifetime: The number of people who matched the audience targeting on the post that clicked anywhere in the post on News Feed, by type. (Unique Users)
post_activity_by_action_type - share post_activity_by_action_type - like post_activity_by_action_type - comment	*Lifetime Post Stories by action type* Action Type: share, like, comment Lifetime: The number of stories created about your Page post, by action type. (Total Count)

Post Impressions:

Metric	***Title & Description***
post_impressions	*Lifetime Post Total Impressions*

	Lifetime: The number of times your Page's post entered a person's screen. Posts include statuses, photos, links, videos and more. (Total Count)
post_impressions_unique	*Lifetime Post Total Reach* Lifetime: The number of people who had your Page's post enter their screen. Posts include statuses, photos, links, videos and more. (Unique Users)
post_impressions_paid	*Lifetime Post Paid Impressions* Lifetime: The number of times your Page's post entered a person's screen through paid distribution such as an ad. (Total Count)
post_impressions_paid_unique	*Lifetime Post Paid Reach* Lifetime: The number of people who had your Page's post enter their screen through paid distribution such as an ad. (Unique Users)
post_impressions_fan	*Lifetime Post Impressions by people who have liked your Page* Lifetime: The number of impressions of your Page post to people who have liked your Page. (Total Count)
post_impressions_fan_unique	*Lifetime Post reach by people who like your Page* Lifetime: The number of people who saw your Page post because they've liked your Page (Unique Users)
post_impressions_fan_paid	*Lifetime Post Paid Impressions by people who have liked your Page* Lifetime: The number of paid impressions of your Page post to people who have liked your Page. (Total Count)
post_impressions_fan_paid_unique	*Lifetime Paid reach of a post by people who like your Page* Lifetime: The number of people who like your Page and who saw your Page post in an ad or sponsored story. (Unique Users)

post_impressions_organic	*Lifetime Post Organic Impressions* Lifetime: The number of times your Page's posts entered a person's screen through unpaid distribution. (Total Count)
post_impressions_organic_unique	*Lifetime Post organic reach* Lifetime: The number of people who had your Page's post enter their screen through unpaid distribution. (Unique Users)
post_impressions_viral	*Lifetime Post Viral Impressions* Lifetime: The number of times your Page's post entered a person's screen with social information attached. Social information displays when a person's friend interacted with you Page or post. This includes when someone's friend likes or follows your Page, engages with a post, shares a photo of your Page and checks into your Page. (Total Count)
post_impressions_viral_unique	*Lifetime Post viral reach* Lifetime: The number of people who had your Page's post enter their screen with social information attached. As a form of organic distribution, social information displays when a person's friend interacted with you Page or post. This includes when someone's friend likes or follows your Page, engages with a post, shares a photo of your Page and checks into your Page. (Unique Users)
post_impressions_nonviral	*Lifetime Post Nonviral Impressions* Lifetime: The number of times your Page's post entered a person's screen. This does not include content created about your Page with social information attached. Social information displays when a person's friend interacted with you Page or post. This includes when someone's friend likes or follows your Page, engages with a post, shares a photo of your Page and checks into your Page. (Total Count)
post_impressions_nonviral_unique	*Lifetime Post Nonviral Reach*

	Lifetime: The number of people who had your Page's post enter their screen. This does not include content created about your Page with social information attached. As a form of organic distribution, social information displays when a person's friend interacted with you Page or post. This includes when someone's friend likes or follows your Page, engages with a post, shares a photo of your Page and checks into your Page. (Unique Users)
post_impressions_by_story_type	*Lifetime Post Viral Impressions by story type* Lifetime: The number of times people saw this post via stories published by their friends. (Total Count)
post_impressions_by_story_type_unique	*Lifetime Post viral reach by story type* Lifetime: The number of people who saw your Page post in a story from a friend, by story type. (Unique Users)

Post Reactions:

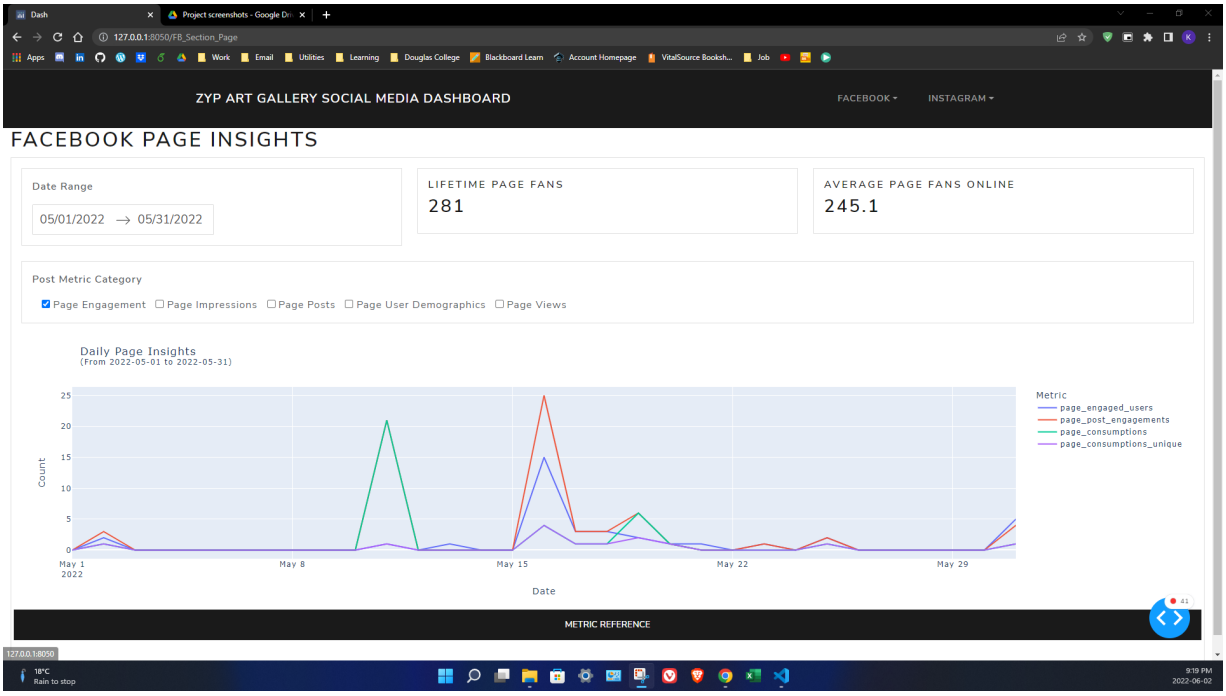
Metric	***Title & Description***
post_reactions_like_total	*Lifetime Total Like Reactions of a post* Lifetime: Total like reactions of a post.
post_reactions_love_total	*Lifetime Total Love Reactions of a post* Lifetime: Total love reactions of a post.
post_reactions_wow_total	*Lifetime Total wow Reactions of a post* Lifetime: Total wow Reactions of a post.
post_reactions_haha_total	*Lifetime Total haha Reactions of a post* Lifetime: Total haha reactions of a post.
post_reactions_sorry_total	*Lifetime Total sad Reactions of a post* Lifetime: Total sad reactions of a post.
post_reactions_anger_total	*Lifetime Total anger Reactions of a post*

	Lifetime: Total anger reactions of a post.

Visualization

As most of the insight related metrics are of period ‘Lifetime’, no aggregations or further calculations were done to create the visualization. Thus, I am taking the metrics at face value. Thus, I believed to best communicate the performance of the Facebook post would be to create a bar chart comparing the metric values to each other. I also thought it would be helpful for a potential user to actually read the entire Facebook post in full form, which is available below the bar chart. This way the user could gauge their own deduction of the metric values more intuitively and organically.

Facebook Page Insights



As can be seen in the above screenshot shows the Facebook Page Insights webpage. The intent of this page is to communicate the performance of the organization’s Facebook page overall.

Interactive elements

As aforementioned, the date range datepicker is consistent element throughout the webpages of the application. On this webpage, it filters the page insight metric values that are within the respective date range on the line chart as per day. The date range that is selected is visible on the line chart title. The line chart’s x-axis starts at the start date and ends at the end date of the date range shown in the datepicker. The datepicker alters the lifetime page fans tile number

based on the end date selected. The datepicker also alters average page fans online tile number based on both the start & end date selected. By default, all the datepickers show a default date range of the last 30 days. The options in the page metric category checklist groups page insight metrics into specific categories; page engagement, impressions, page posts, user demographics, and page views.

Data

This webpage uses the “ZypFacebook_Insights1”, “ZypFacebook_Insights2”, and “ZypFacebook_Insights3” Google Sheets document that are saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

The following refers to information visualized on the lifetime page fans tile.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans	*Lifetime Total Likes* Lifetime: The total number of people who have liked your Page. (Unique Users)

The following refers to information visualized on the average page fans online tile.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_online_per_day	*Daily liked and Online by Day* Daily: The number of people who liked your Page and who were online on the specified day. (Unique Users)

The following information is visualized on the line chart.

Field	***Description***
end_time	Date

Page Engagment:

Metric	***Title & Description***
page_engaged_users	*Daily Page Engaged Users* Daily: The number of people who engaged with your Page. Engagement includes any click or story created. (Unique Users)
page_post_engagements	*Daily Post Engagements* Daily: The number of times people have engaged with your posts through like, comments and shares and more.
page_consumptions	*Daily Page Consumptions* Daily: The number of clicks on any of your content. Stories generated without clicks on page content (e.g., liking the page in Timeline) are not included. (Total Count)
page_consumptions_unique	*Daily Total Consumers* Daily: The number of people who clicked on any of your content. Stories that are created without clicking on Page content (ex, liking the Page from timeline) are not included. (Unique Users)

Page Impressions:

Metric	***Title & Description***
page_impressions	*Daily Total Impressions* Daily: The number of times any content from your Page or about your Page entered a person's screen. This includes posts, stories, ads, as well other content or information on your Page. (Total Count)
page_impressions_unique	*Daily Total Reach* Daily: The number of people who had any content from your Page or about your Page enter their screen. This includes posts, check-ins, ads, social information from people who interact with your Page and more. (Unique Users)

Page Posts Impressions (i.e., Page Posts):

Metric	***Title & Description***
page_posts_impressions	*Daily Total Impressions of your posts*

	Daily: The number of times your Page's posts entered a person's screen. Posts include statuses, photos, links, videos and more. (Total Count)
page_posts_impressions_unique	*Daily Reach Of Page Posts* Daily: The number of people who had any of your Page's posts enter their screen. Posts include statuses, photos, links, videos and more. (Unique Users)

Page User Demographics:

Metric	***Title & Description***
page_fan_adds	*Daily New Likes* Daily: The number of new people who have liked your Page (Total Count)
page_fan_adds_unique	*Daily New Likes Unique* Daily: The number of new people who have liked your Page (Unique Users)
page_fan_removes	*Daily Unlikes* Daily: The number of Unlikes of your Page (Total Count)
page_fan_removes_unique	*Daily Unlikes Unique* Daily: The number of Unlikes of your Page (Unique Users)

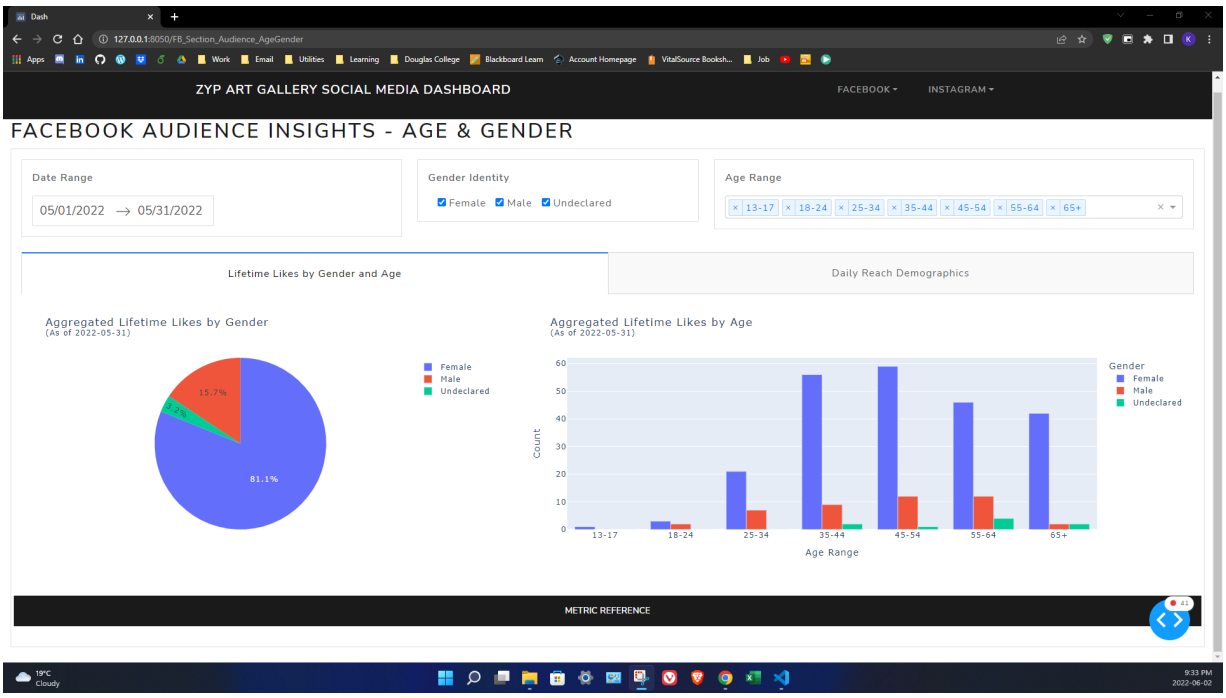
Page Views:

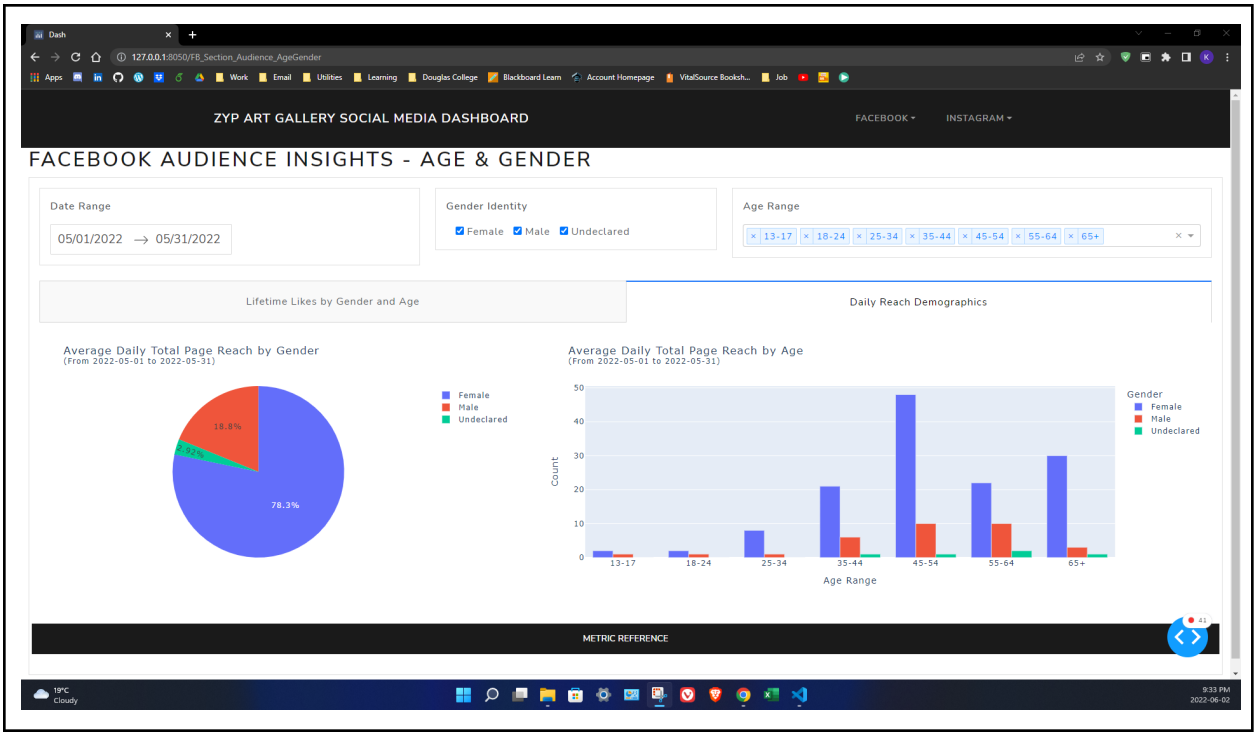
Metric	***Title & Description***
page_views_total	*Daily Total views count per Page* Daily: Total views count per Page
page_views_logged_in_total	*Daily Total logged-in views count per Page* Daily: Total logged-in views count per Page
page_views_logged_in_unique	*Daily Total logged-in views count per Page* Daily: Total logged-in views count per Page (Unique Users)

Visualization

As there are many page insight metrics that are of the period “Daily”, I felt it would be interesting to track these metrics over a period of time (i.e. per day) to assess the performance fluctuations of the Facebook page. Specifically, the “ZypFacebook_Insights1” Google Sheets data file was used to create the line chart. The lifetime page fans tile utilizes the “ZypFacebook_Insights3” Google Sheets Data file. It is of period “Lifetime”, so it simply takes the metric value that is as of the end date selected on the date range datepicker. Furthermore, there is no need to perform aggregations or further calculations. However, as the average page fans online is of period “Daily”, it would make sense to perform an average aggregation of metric values from the start and end date displayed on the date range datepicker. The “ZypFacebook_Insights2” Google Sheets data file is used to calculate the number in the average page fans online tile.

Facebook Audience Insights - Age & Gender





As can be seen in the above screenshot shows the Facebook Audience Insights - Age & Gender webpage. The overall intent of this webpage is to assess the performance of the Facebook page by age and gender.

Interactive elements

There are two tabs. The intent of the first tab is to communicate the likes of the Facebook page by Age & Gender. The metric used in this tab is of period “Lifetime”. Thus, the visualizations change based on the end date selected on the date range datepicker. The intent of the second tab it to communicate the reach of the Facebook page by Age & Gender. The metric used in this tab is of period “Daily”. Thus, the visualizations change based on the start & date selected on the date range picker, and calculates the average of the metric value based on date range. The titles of the visualizations in both tabs change based on the selections made in the datepicker. The gender identity checklist and age range dropdowns also filter and adjust the visualizations accordingly. If a gender option or age range option is de-selected, the visualizations would not display them, and associate them with any calculation or aggregation leading up to rendering the charts. Although, by default all gender and age options are selected.

Data

This webpage uses the “ZypFacebook_Audience-Age&Gender1” and “ZypFacebook_Audience-Age&Gender2” Google Sheets documents that are saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

The following refers to information visualized on the ‘Lifetime Likes by Gender and Age’ tab.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_gender_age [Gender abbreviation].[Age interval] E.g. F.13-17 (Female aged 13 to 17) ● F = Female ● M = Male ● U = Unknown	*Lifetime Likes by Gender and Age* Lifetime: Aggregated demographic data about the people who like your Page based on the age and gender information they provide in their user profiles. (Unique Users)

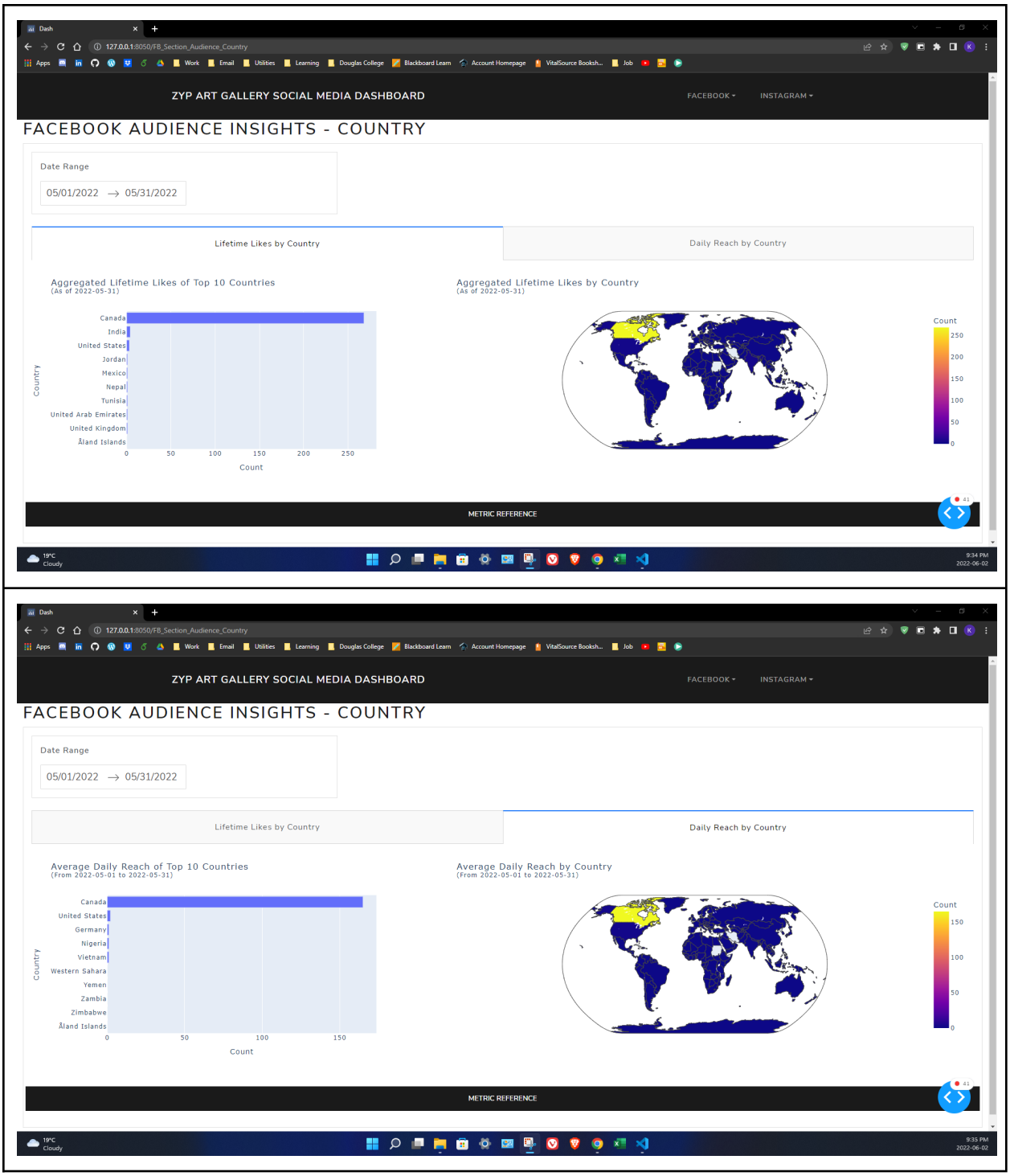
The following referes to information visualized on the ‘Daily Reach Demographics’ tab.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_impressions_by_age_gender_unique [Gender abbreviation].[Age interval] => E.g. F.13-17 (Female aged 13 to 17) ● F = Female ● M = Male ● U = Unknown	*Daily Reach Demographics* Daily: Total Page Reach by age and gender. (Unique Users)

Visualization

Regardless of which tab is in view, there are two charts. The pie chart shows the percentage proportion of the metric by gender identity. The side-by-side bar chart compares the metric values by gender across a set of age ranges. The ‘Lifetime Likes by Gender & Age’ tab uses the “ZypFacebook_Audience-Age&Gender1” Google Sheet data file, whilst the ‘Daily Reach Demographics’ tab uses the “ZypFacebook_Audience-Age&Gender2” Google Sheet data file.

Facebook Audience Insights - Country



As can be seen in the above screenshot shows the Facebook Audience Insights - Country webpage. The overall intent of this webpage is to assess the performance of the Facebook page by country.

Interactive elements

There are two tabs. The intent of the first tab is to communicate the likes of the Facebook page by Country. The metric used in this tab is of period “Lifetime”. Thus, the visualizations change based on the end date selected on the date range datepicker. The intent of the second tab is to communicate the reach of the Facebook page by Country. The metric used in this tab is of period “Daily”. Thus, the visualizations change based on the start & date selected on the date range picker, and calculates the average of the metric value based on date range. The titles of the visualizations in both tabs change based on the selections made in the datepicker.

Data

This webpage uses the “ZypFacebook_Audience-Country1” and “ZypFacebook_Audience-Country2” Google Sheets documents that are saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

The following refers to information visualized on the ‘Lifetime Likes by Country’ tab.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_country Country name => E.g. South Africa	*Lifetime Likes by Country* Lifetime: Aggregated Facebook location data, sorted by country (top 50), about the people who like your Page. (Unique Users)

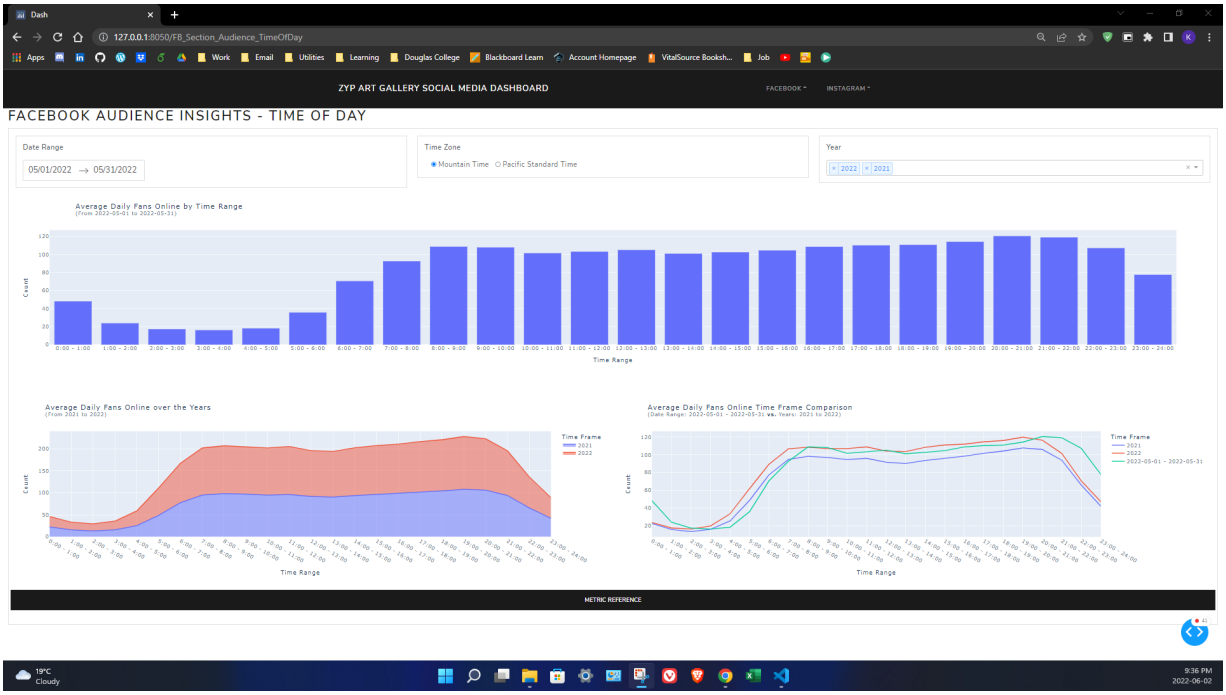
The following referes to information visualized on the ‘Daily Reach by Country’ tab.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_impressions_by_country_unique Country name => E.g. South Africa	*Daily Reach by Country* Daily: Total Page Reach by user country. (Unique Users)

Visualization

Regardless of which tab is in view, there are two charts. The bar chart shows the top countries regarding the metric value. The choropleth bar chart shows the metric value of all countries in the form of a map. The ‘Lifetime Likes by Country’ tab uses the “ZypFacebook_Audience-Country1” Google Sheet data file, whilst the ‘Daily Reach by Country’ tab uses the “ZypFacebook_Audience-Country2” Google Sheet data file.

Facebook Audience Insights - Time of Day



As can be seen in the above screenshot shows the Facebook Audience Insights - Time of Day webpage. The overall intent of this webpage is to assess the performance of the Facebook page by time of day.

Interactive elements

The date range datepicker changes the bar chart visualization and line chart visualization. This is specifically so as the aforementioned visualizations calculate the average of the page fans online based on the start and end date of the datepicker. The radio button selector showing the time zones changes all visualizations on the page. The Facebook data is extracted as per Pacific Standard Time by default. Yet as Zyp Art Gallery is an organization based in Alberta that follows Mountain Time, I decided to provide an option to a user to toggle between both time zones and Mountain Time is set to default. Thus the average page fans online correspond to Mountain Time time ranges if the Mountain Time option is selected. The year dropdown alters the area chart visualization and line chart visualization by including areas and lines respectively that indicate average page fans online per time range.

Data

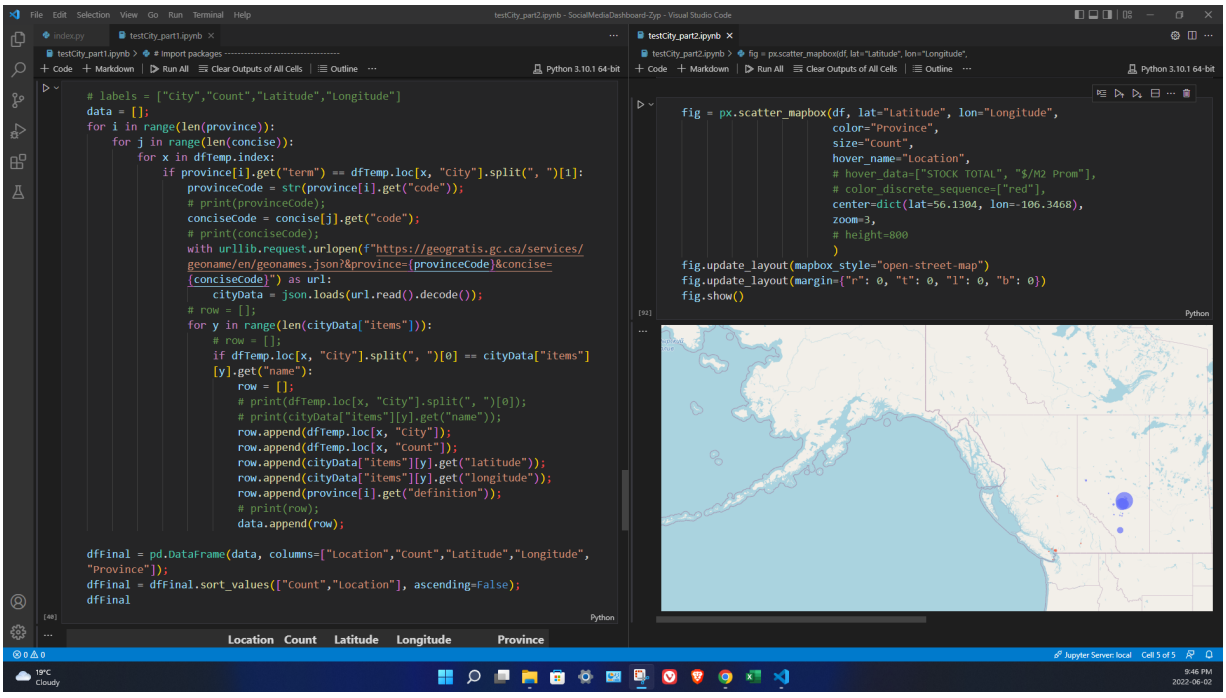
This webpage uses the “ZypFacebook_Audience-TimeOfDay” Google Sheet document that are saved on the Organization’s Google Drive. The following is a breakdown of the fields and metrics that are used on this webpage.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_online Time range in the 24 hour format => E.g. 14:00 - 15:00	*Daily Liked and Online* Daily: The number of people who liked your Page and when they are online in PST/PDT (Unique Users)

Visualization

The bar chart visualization shows the the average page fans online by time range within the start and end date selected in the date range datepicker. The area chart visualization shows the average page fans online over the years. The area chart calculates the average page fans online per the years selected in the year dropdown. The line chart visualization compares the average page fans online within the date range to the current & prior annual years based on the years selected in the respective dropdown.

Facebook Audience - Canadian City (in progress)



I began coding the Facebook Canadian City webpage; evidence shown above. I am currently using the “ZypFacebook_Audience-CanadianCity1” Google Sheets data file. It took me quite long I was experimenting how to best clean and aggregate the data. The data file actually had 9026 columns so it took too long to simulate selection of a date, and then clean & aggregate based on selection of that date. As the data file does not have longitude and latitude values of

the Canadian City locations, I needed to retrieve them so that I could attempt to plot a scatter bubble map. I used the Geoname API from the Canadian government website as the source to retrieve the longitude and latitude values. I faced several challenges with using the API as I encountered HTTP errors sometimes. Also running this specific piece of code used to take more than five minutes for some reason. It could be due to the number of columns of the data file and performing several API requests in a loop to construct a final dataframe to construct a bubble scatter map chart. I eventually did manage to create a demo version of the visualization but only time will tell using the Canadian City data files are ideal given that data processing takes long. In a perfect world, the data processing will not take too long once deployed on Heroku but may take long on a local machine. The visualization in particular shows different colored bubbles based on Province and different sized bubbles based on metric value.

The following is a breakdown of the fields and metrics that are used when prototyping the prospective Canadian City webpage for the Facebook section.

Field	***Description***
end_time	Date
Metric	***Title & Description***
page_fans_city City name, Province abbreviation, Country => E.g. Calgary, AB, Canada	*Lifetime Likes by City* Lifetime: Aggregated Facebook location data, sorted by city (top 50), about the people who like your Page. (Unique Users)

Pending Implementations

- Facebook Audience insights - Canadian City
- Instagram section
    - User interface/layout design
    - Visualization exploration
    - Coding and internal testing
- Landing page
    - User interface/layout design
    - Potential content
    - Coding and internal testing
- Authentication
    - Further research
    - Coding and internal testing
- Deployment
    - Research
    - Practice and testing

Proposed Revisions

I am currently somewhat on track in regards to completing my project code wise. I feel I could potentially add more interactive features in the Facebook Country webpage. Perhaps a continent filter that adjusts the visualizations. Specifically, selecting a continent would focus the choropleth to the selected continent, and then show countries in the bar chart that only reflect the selected continent. This would mean I would have to adjust the cleaning and aggregation code leading up to the country-related visualizations. I am considering doing something similar to this but for the Facebook Canadian City webpage. All of this depends on practically and availability of time.