Final Project

Logic Apps - Social Media Monitoring

Prouty, Stephen

Deep Azure@McKesson
Dr. Zoran B. Djordjević

Problem Statement

Social media applications have become part of our everyday lives. This is even more true for the younger generations. Parents are frequently left blind to the messages being sent by their children. They need a way to monitor the usage of these messaging applications to ensure appropriate and safe usage. The goal of this project is to capture messages from these applications and load them into a central location where the information can be viewed by a parent.

Introduction to Azure Logic Apps

"Logic Apps helps you build, schedule, and automate processes as workflows so you can integrate apps, data, systems, and services across enterprises or organizations. Logic Apps simplifies how you design and create scalable solutions for app integration, data integration, system integration, enterprise application integration (EAI), and business-to-business (B2B) communication, whether in the cloud, on premises, or both."

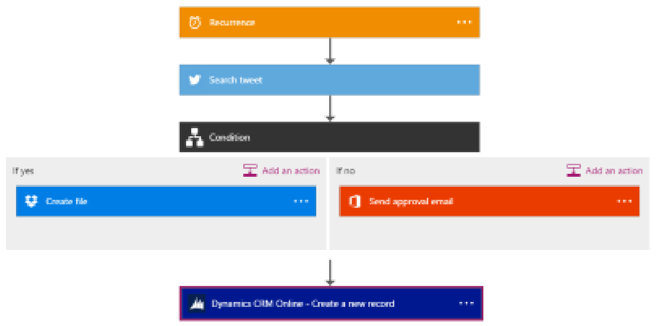
https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-overview

Why Logic Apps?

- Easily and quickly connect systems using prebuilt APIs (~200 currently available)
- Allow developers to focus on applications instead of infrastructure
- Services automatically scale to meet application needs
- Many implementations don't require coding
- Seamlessly integrate with Function Apps for custom actions
- Consumption based pricing (https://azure.microsoft.com/en-us/pricing/details/logic-apps/)

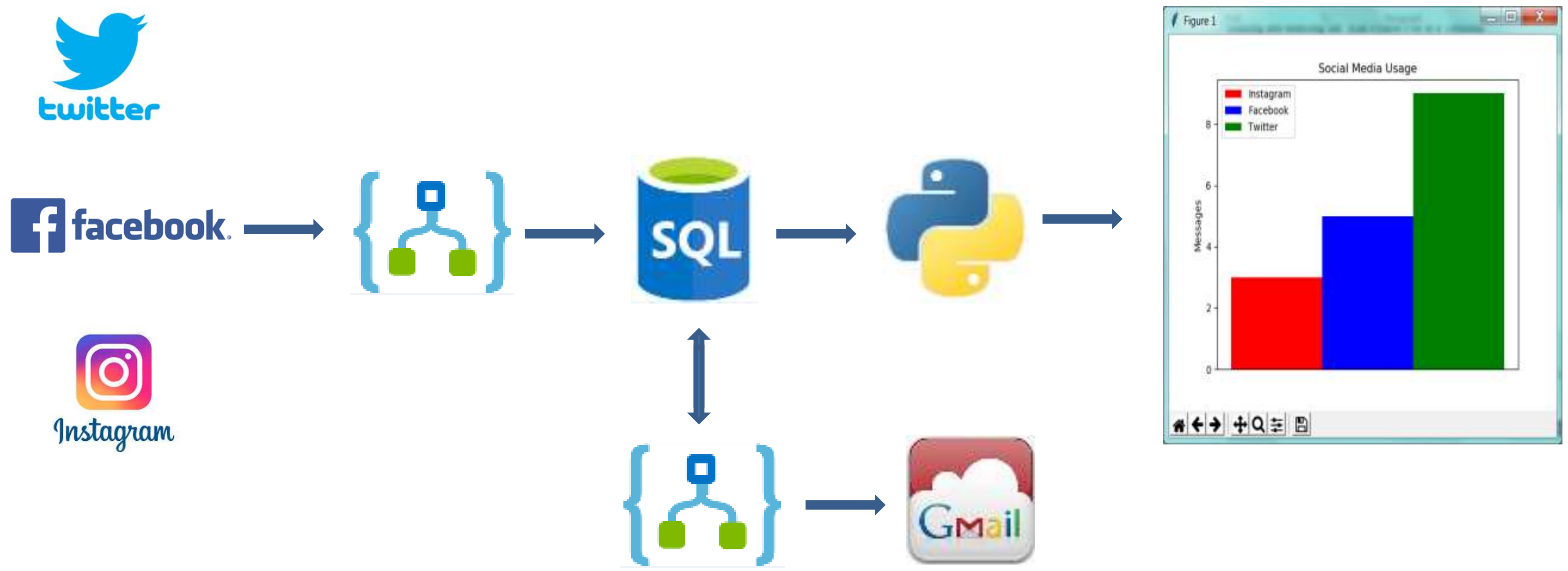
High-Level How Logic Apps Work

- A Logic App workflow starts with a trigger which fires when a specific event occurs
- The Logic Apps engine creates a Logic App instance and runs the workflow actions



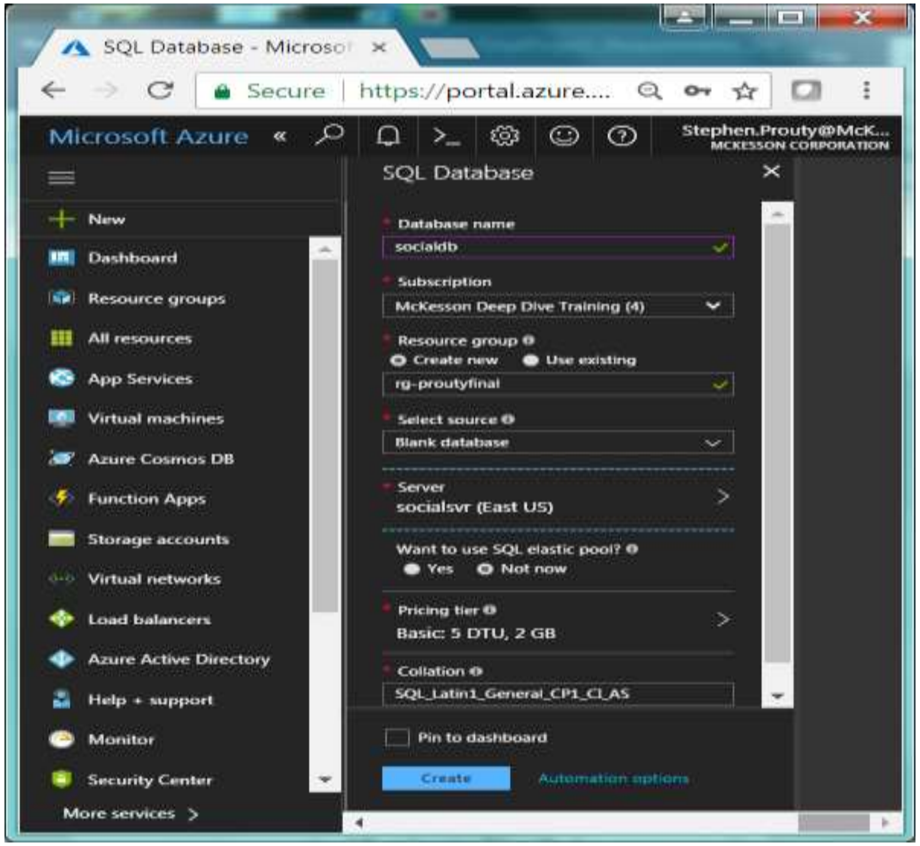
Project High Level Overview

1. Azure SQL database to store social media messages
2. Logic Apps for three API connectors: Instagram, Facebook, Twitter
3. Function App to aggregate the message data in the Azure SQL database
4. Logic App to call Function App and send notification via email
5. Python script to graph summary message data from Azure SQL database



Step 1 – Create Azure SQL Database

One of the requirements for this problem solution was to centralize all incoming social media messages into a central repository. This central repository needed to be easily accessible by the end user for reviewing messages. The Azure SQL database was selected to fulfill this requirement.



SQL Database

Database name: socialdb

Subscription: McKesson Deep Dive Training (4)

Resource group: Create new [✓ Use existing] rg-proutyfinal

Select source: Blank database

Server: socialsvr (East US)

Want to use SQL elastic pool? Yes [✓ Not now]

Pricing tier: Basic: 5 DTU, 2 GB

Collation: SQL_Latin1_General_CP1_CI_AS

Pin to dashboard

MESSAGE_DETAIL

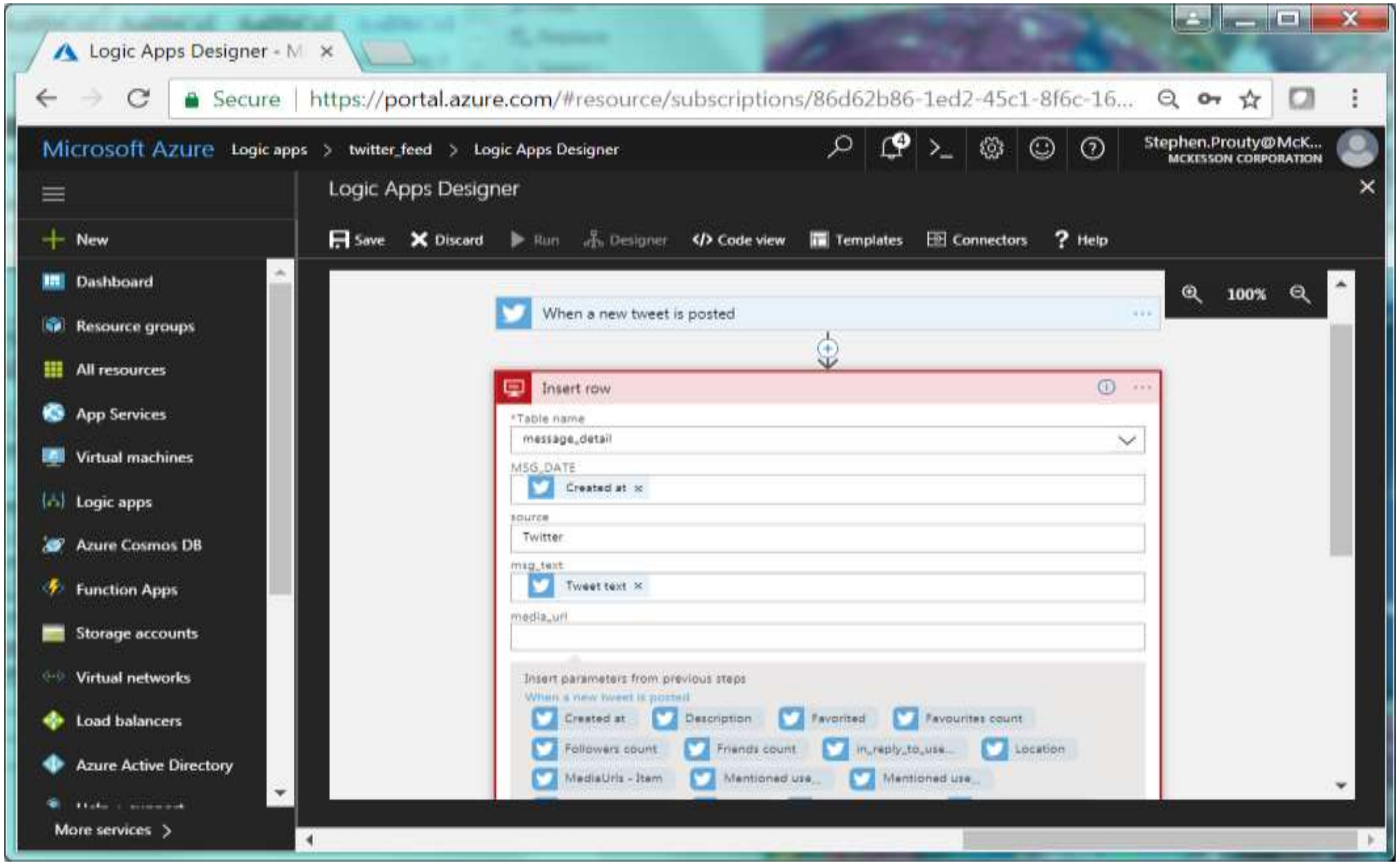
MSG_DATE	DATE
SOURCE	TEXT
MSG_TEXT	TEXT

MESSAGE_SUMMARY

SUMMARY_DATE	DATE
SOURCE	TEXT
MSG_COUN	INTEGER

Step 2 – Create Social Media API Logic

Logic Apps provides ~200 built-in connectors that provide the ability to trigger based on events.   The Twitter, Facebook, and Instagram built-in connectors will be used in this project to trigger when a new message is posted to the application. These trigger events will then utilize an Azure SQL connector to execute a “Insert Row” action.



When a new tweet is posted

Insert row

*Table name: message_detail

MSG_DATE: Created at

source: Twitter

msg_text: Tweet text

media_url:

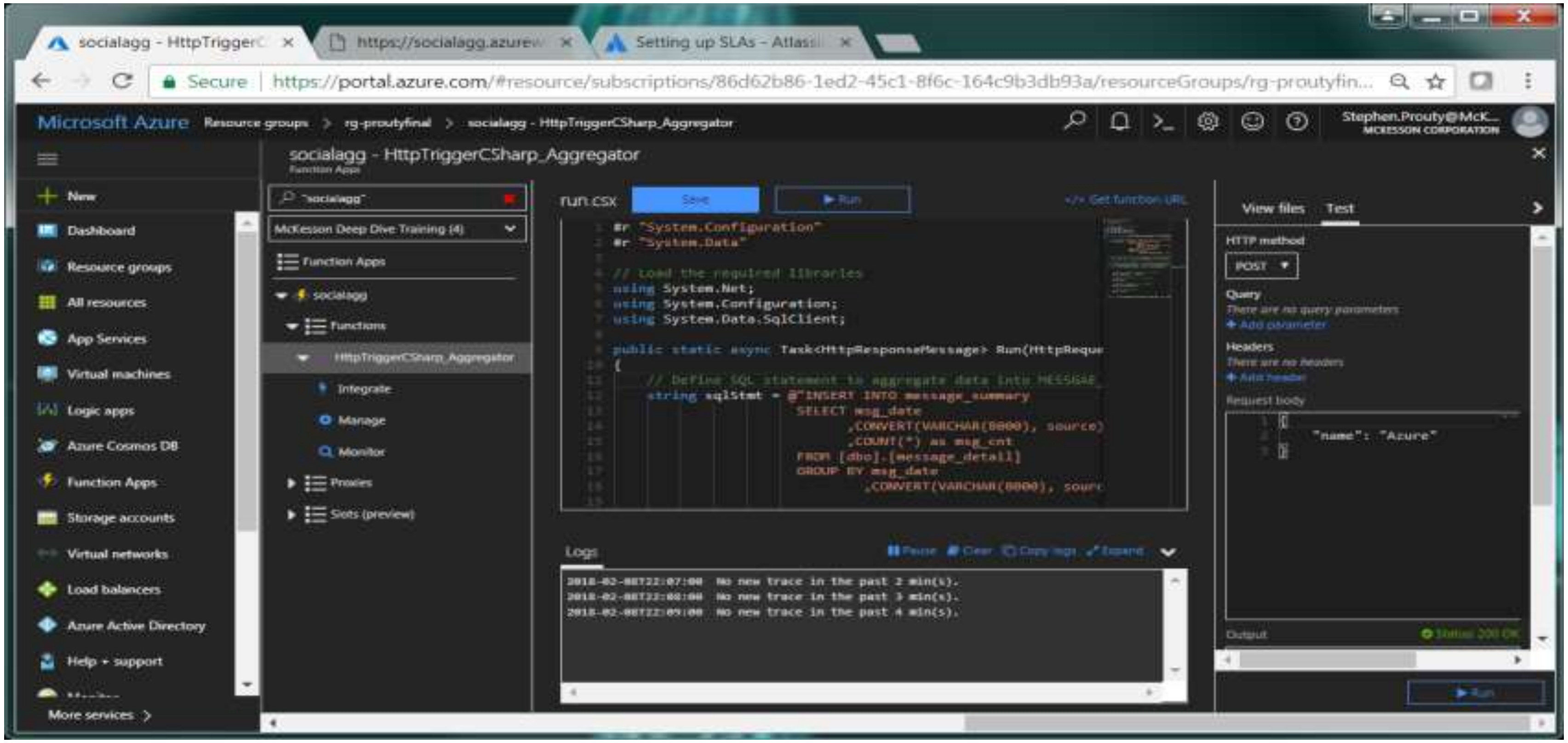
Insert parameters from previous steps

When a new tweet is posted

Created at · Description · Favorited · Favourites count · Followers count · Friends count · in_reply_to_use... · Location · MediaUrls - Item · Mentioned use... · Mentioned use...

Step 3 – Create Function to Aggregate Data

One of the goals of this project was to produce summarized data showing the number of messages posted to each social media application. In order to meet this goal, the detailed messages being extracted from the social media sites needed to be aggregated into an easy to query summarized table. This action was accomplished by programming a C# Function App.



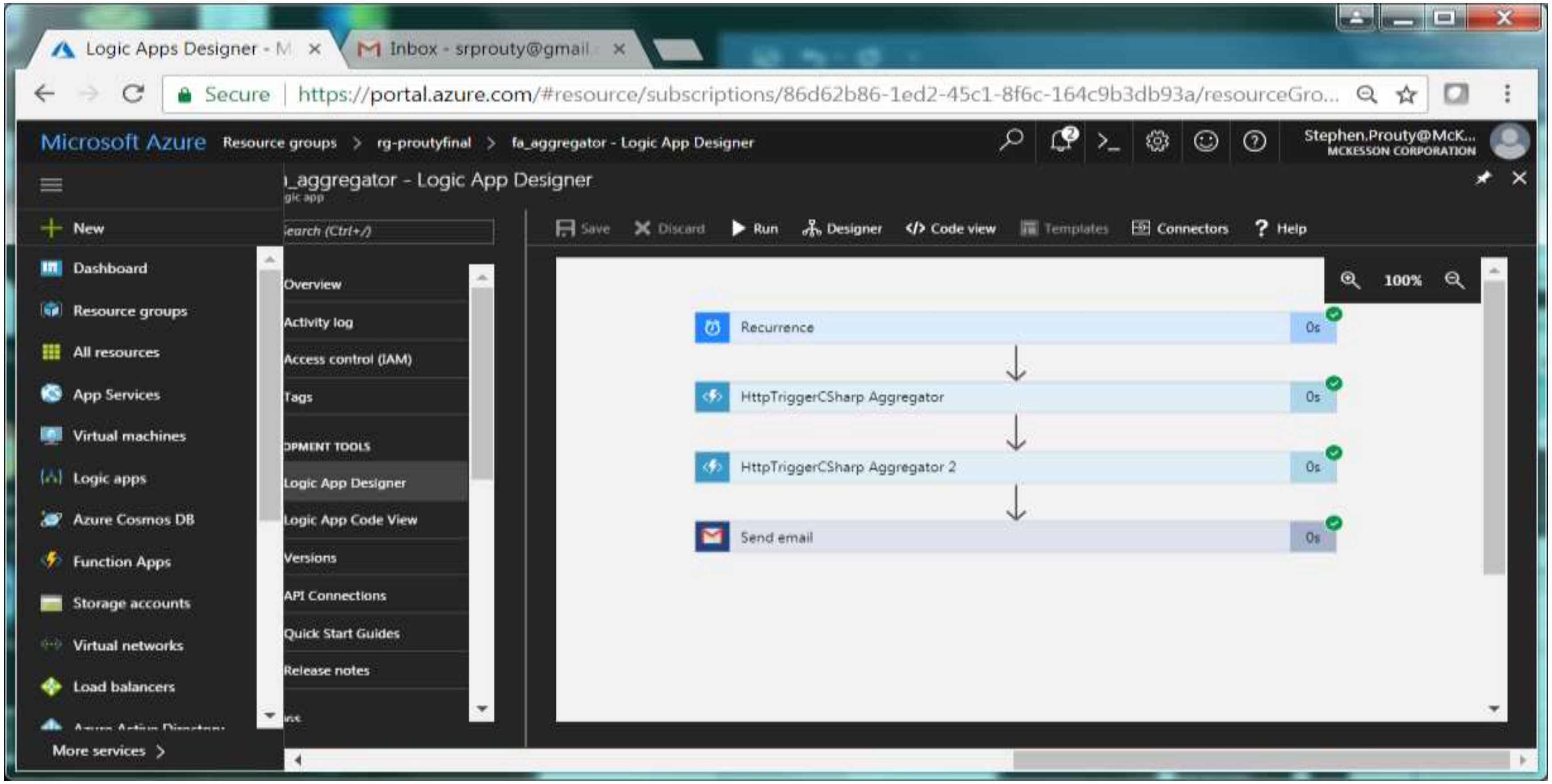
```
#r "System.Configuration"
#r "System.Data"

// Load the required libraries
using System.Net;
using System.Configuration;
using System.Data.SqlClient;

public static async Task<HttpResponseMessage> Run(HttpReque
{
    // Define SQL statement to aggregate data into MESSAGE_
    string sqlStmt = @"INSERT INTO message_summary
                       SELECT msg_date
                              ,CONVERT(VARCHAR(8000), source)
                              ,COUNT(*) as msg_cnt
                       FROM [dbo].[message_detail]
                       GROUP BY msg_date
                                ,CONVERT(VARCHAR(8000), sourc
```

```
2018-02-08T22:07:00  No new trace in the past 2 min(s).
2018-02-08T22:08:00  No new trace in the past 3 min(s).
2018-02-08T22:09:00  No new trace in the past 4 min(s).
```

```
{
    "name": "Azure"
}
```

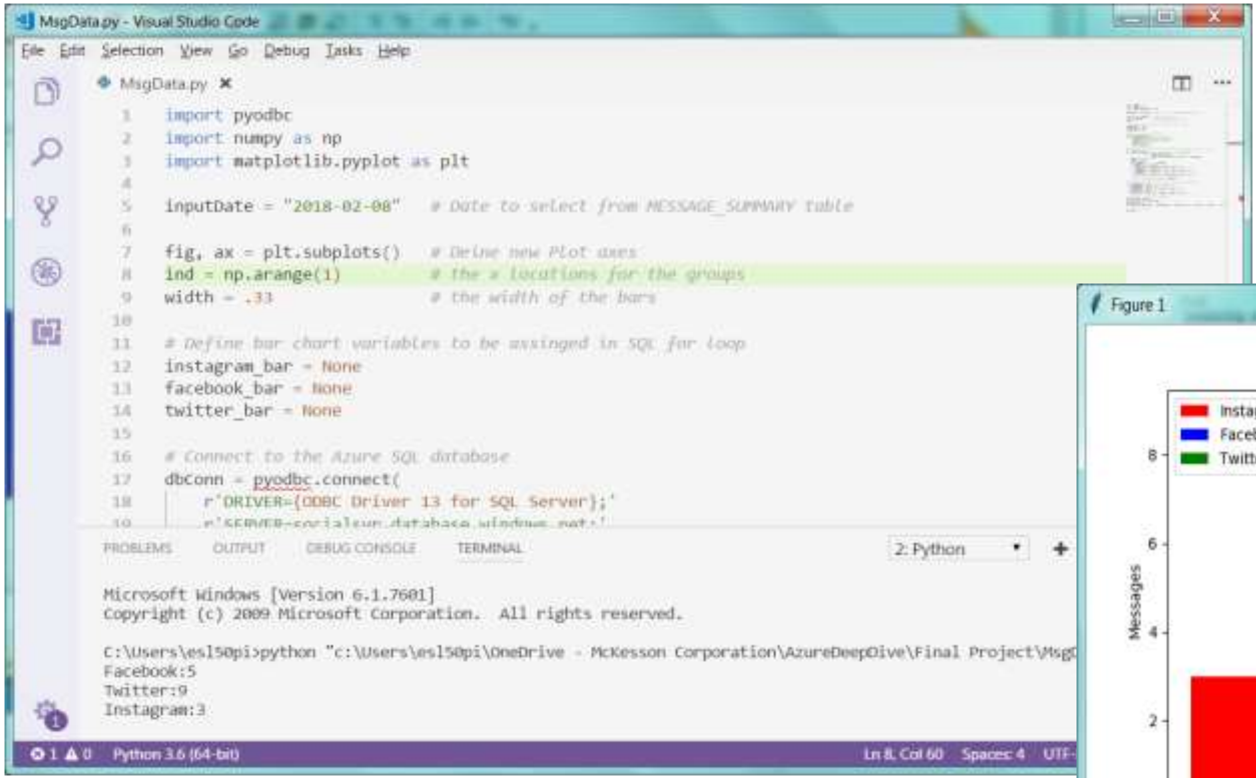
Step 4 – Create Logic App to Call Aggregation Function

In addition to ingesting data, Logic Apps also have the ability to send outbound messages. For this application, the Logic App will first aggregate the data using the previously created Function and then send a notification via GMail to the user that the data is ready to be viewed.



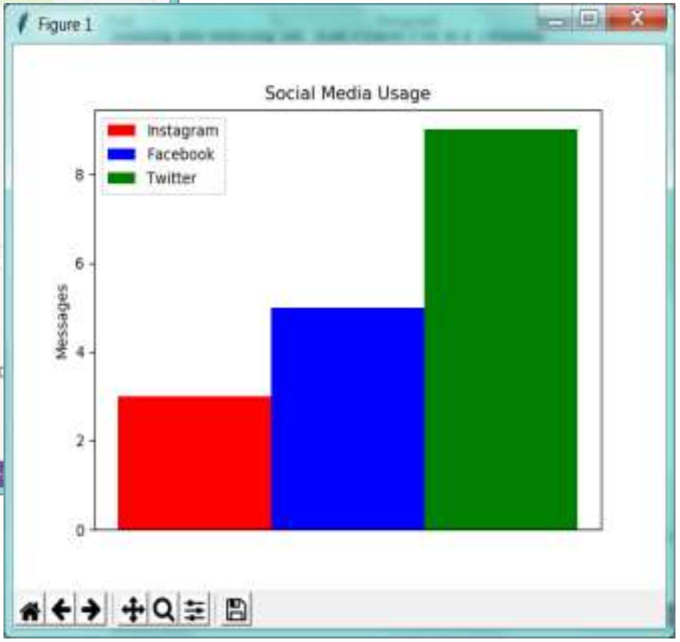
Step 5 – Create Python Script to Graph Data

The final step for this project generates a bar chart displaying the message counts for each of the social media applications. In order to accomplish this task, a Python client script utilizing the MatPlotLib module was used.


MsgData.py - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
MsgData.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: Python
Python 3.6 (64-bit)
Ln 8, Col 60 Spaces: 4

```python
import pyodbc
import numpy as np
import matplotlib.pyplot as plt

inputDate = "2018-02-08"   # Date to select from MESSAGE_SUMMARY table

fig, ax = plt.subplots()   # Define new Plot axes
ind = np.arange(1)         # the x locations for the groups
width = .33                # the width of the bars

# Define bar chart variables to be assinged in SQL for loop
instagram_bar = None
facebook_bar = None
twitter_bar = None

# Connect to the Azure SQL database
dbConn = pyodbc.connect(
    r'DRIVER={ODBC Driver 13 for SQL Server};'
    r'SERVER=socialsvr.database.windows.net;'
```

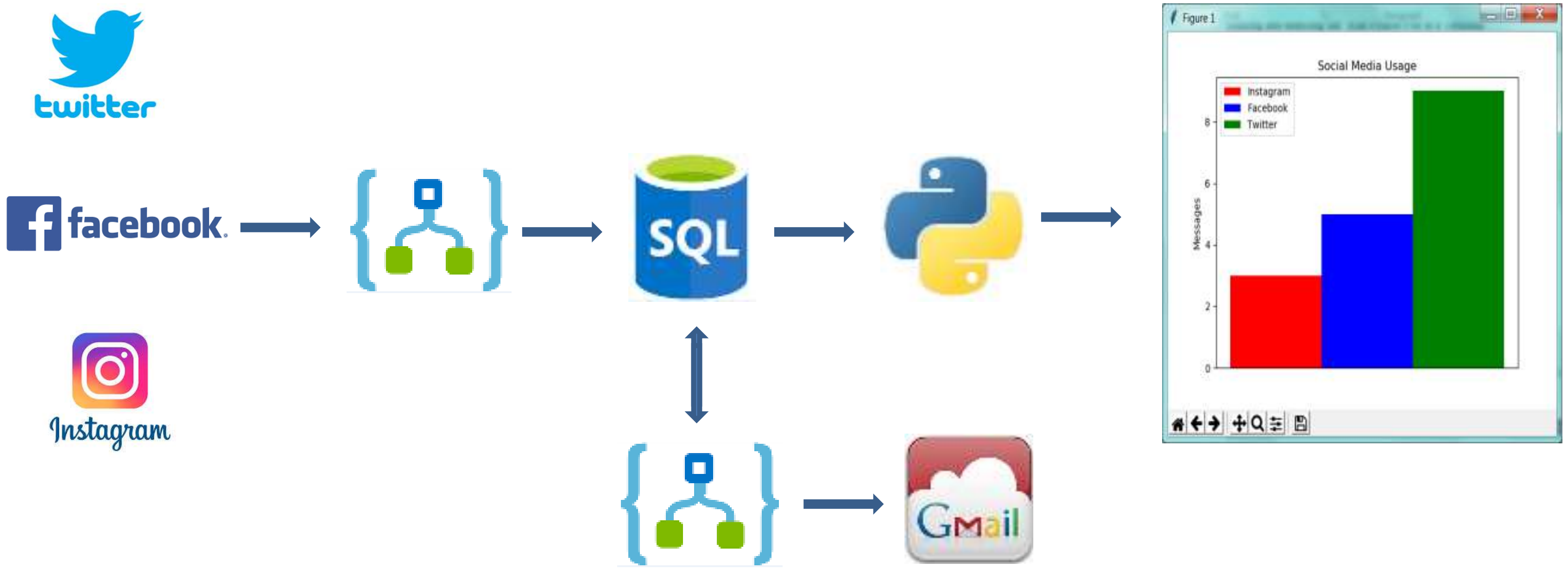
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\esl50pi>python "c:\Users\esl50pi\OneDrive - McKesson Corporation\AzureDeepDive\Final Project\Msg
Facebook:5
Twitter:9
Instagram:3
```



Demo

The demo will:

- Post a few messages to the social media applications
- Execute the Logic Apps to retrieve the messages and load to the DB
- Execute the Logic App to aggregate the data
- Run the Python script to produce a bar chart of the data



Lessons Learned

- Logic Apps is a very flexible architecture capable of delivering complex workflow
- In many cases, there is minimal coding required
- Applications can be easily customized using built in workflow features
  - Parallel processing
  - Conditional logic
- Additional, more complex, logic can be added using Funcions
- Logic Apps currently supports ~200 different connection APIs
  - Microsoft resources: blob storage, SQL Server, event grids, …
  - External resources: GMail, Twitter, Facebook, HipChat, …

References

Two minute (short): https://youtu.be/Nq21domeXjc

15 minutes (long): https://youtu.be/Hf0IBo7nDko

GitHub Repository: https://github.com/scmprouty/AzureDeepFinal