

Assignment 4

Part B

Meet Patel (B00899516)

Dalhousie University

Subject

CSCI 5410 (Serverless Data Processing)

Professor

Dr. Saurabh Dey

Project Git Repository

Gitlab Repository Link: https://git.cs.dal.ca/patel13/csci5410_b00899516_meet_patel.git

Screenshots of all the steps performed

Initial Google Cloud Platform Setup

Figure 1 to 11 shows the prerequisite steps that are required to do build this application. These steps are performed on google cloud platform.

Figure 1 shows the creation of new project on GCP with name “**CSCI5410-Assignment4-PartB**”.

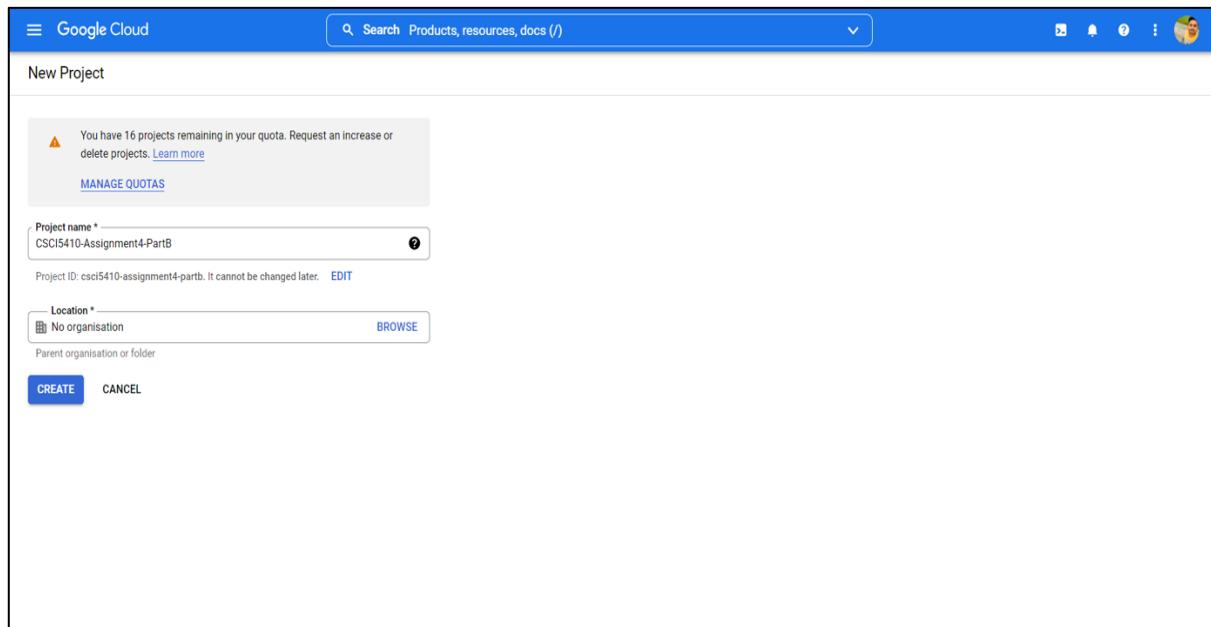


Figure 1: Creation of CSCI5410-Assignment4-PartB in Google Cloud Platform (GCP)

Figure 2 shows the dashboard of the newly created project “**CSCI5410-Assignment4-PartB**”

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516

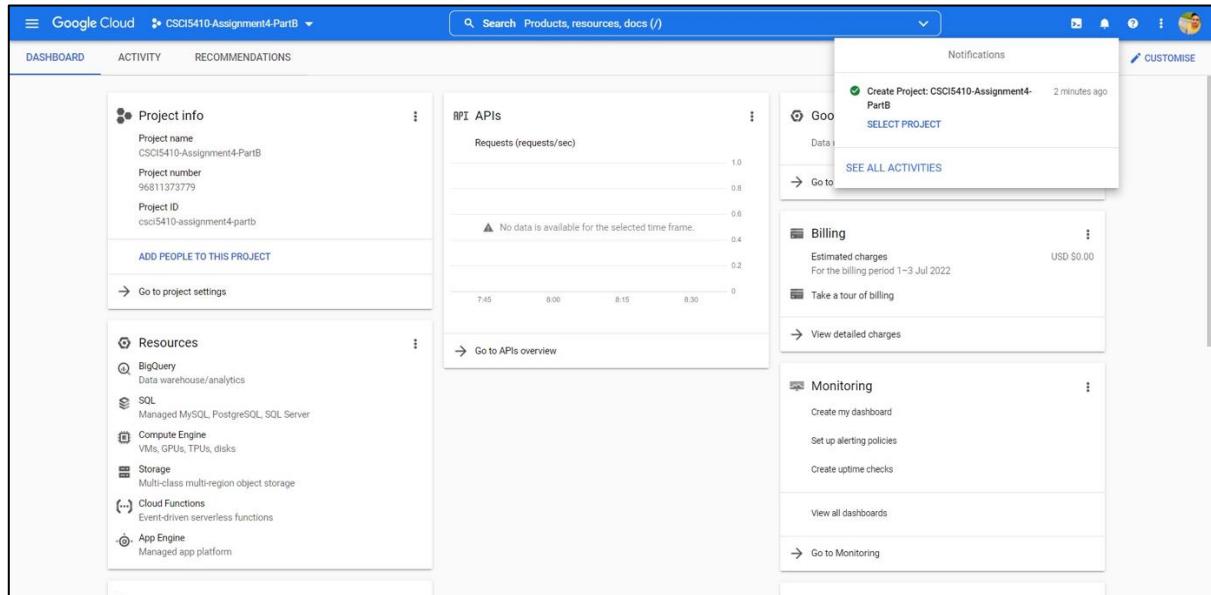


Figure 2: Dashboard of the newly created project "CSCI5410-Assignment4-PartB"

Once the creation of new project on google cloud platform is completed, we need to create a service account on GCP which helps us to connect our application with various other google cloud services.

Figure 3 to 5 represents the steps that are required to create a new service account with name "**CSCI5410-Assignment4-PartB-SA**" Inside the google cloud project with name "**CSCI5410-Assignment4-PartB**"

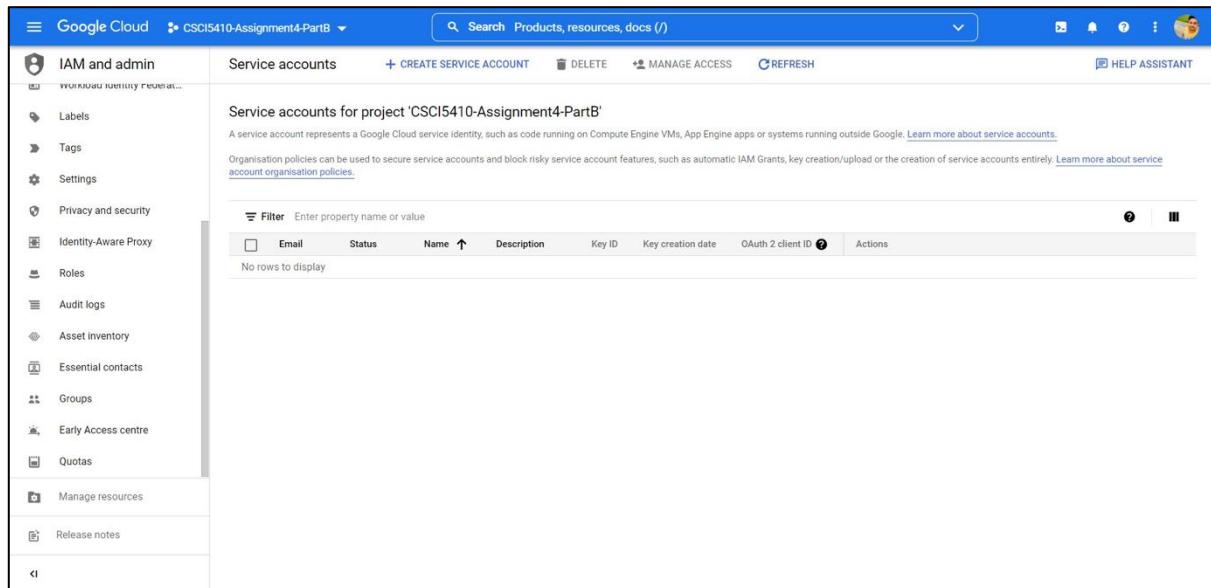


Figure 3: Creation of service account for GCP project "CSCI5410-Assignment4-PartB"

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516

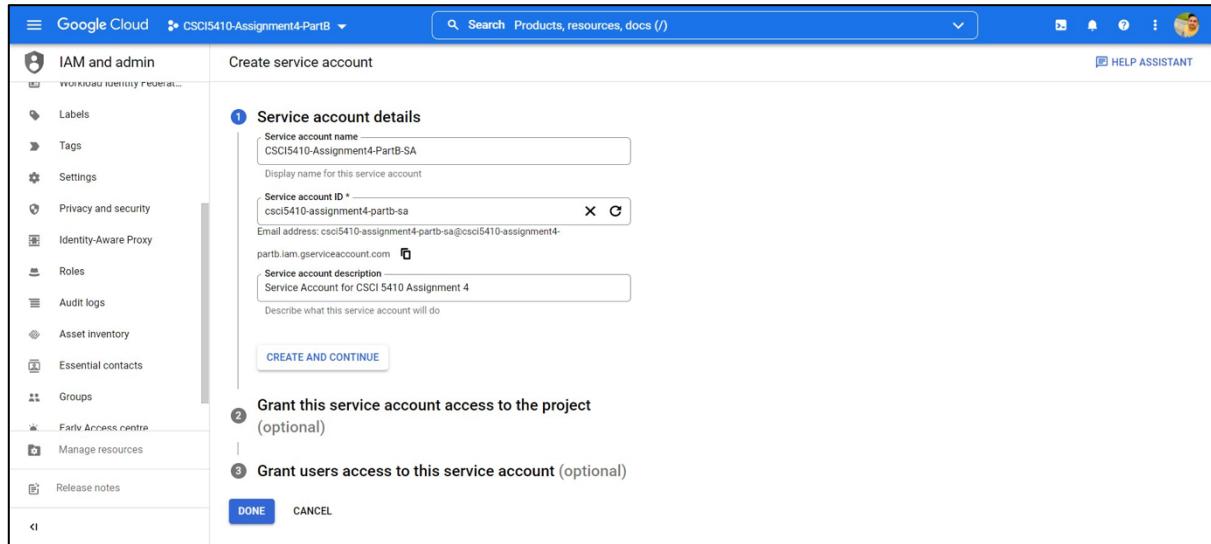


Figure 4: Creation of service account for GCP project "CSCI5410-Assignment4-PartB"

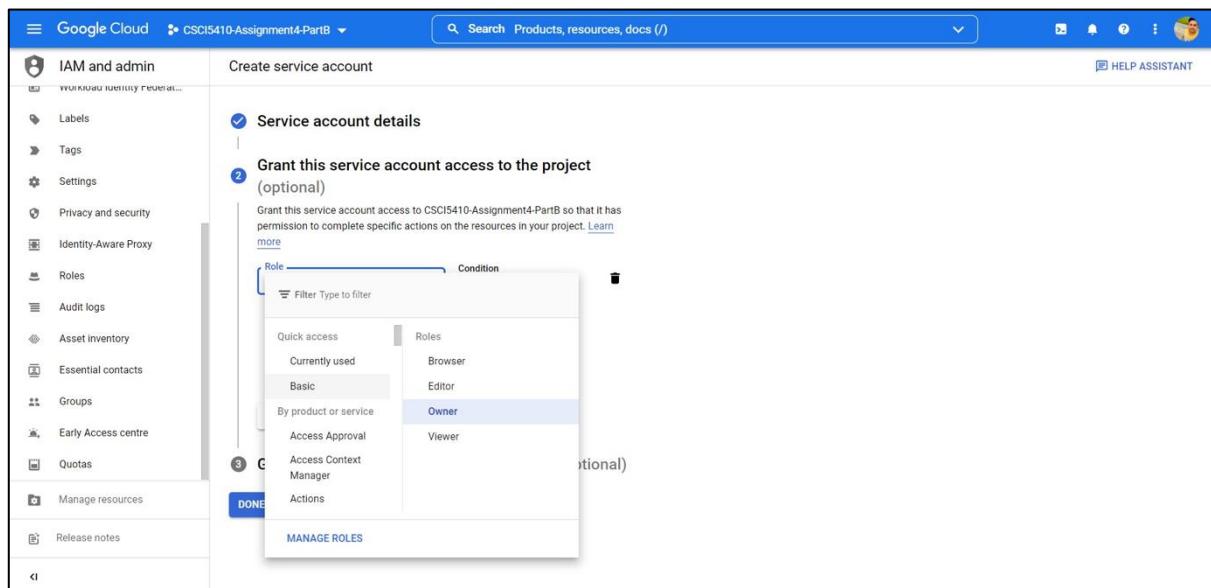


Figure 5: Creation of service account for GCP project "CSCI5410-Assignment4-PartB"

The screenshot shows the Google Cloud IAM and admin interface. On the left, there's a sidebar with various options like IAM and admin, Labels, Tags, Settings, Privacy and security, Identity-Aware Proxy, Roles, Audit logs, Asset inventory, Essential contacts, Groups, Early Access centre, Manage resources, and Release notes. The main area is titled 'Service accounts' and shows a table with one row. The table columns are Email, Status, Name, Description, Key ID, Key creation date, OAuth 2 client ID, and Actions. The single entry is 'csci5410-assignment4-partb-sa@csci5410-assignment4-partb.iam.gserviceaccount.com' with a green checkmark in the Status column. The 'Actions' column contains a three-dot menu icon.

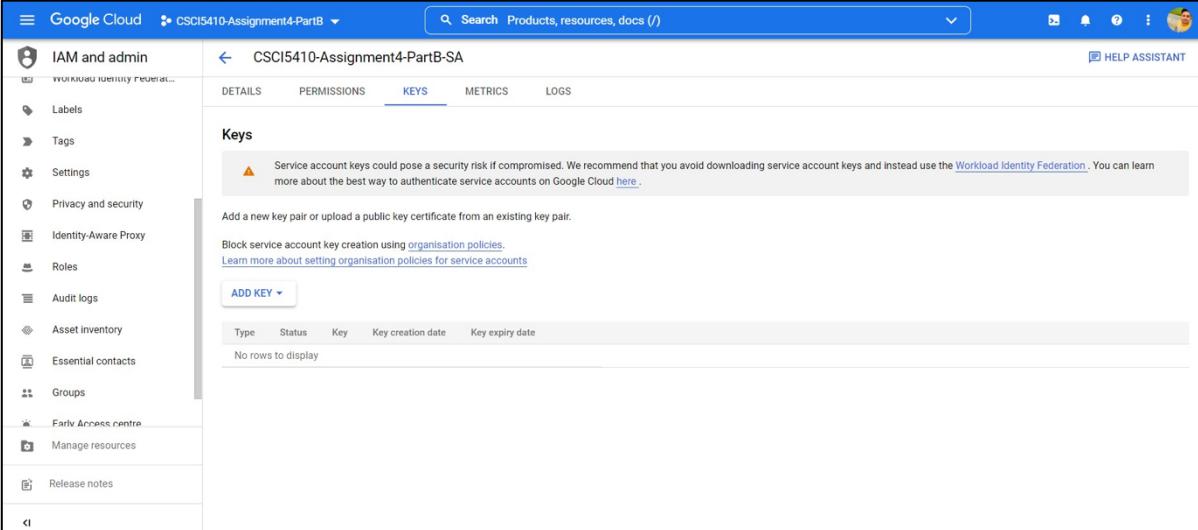
Figure 6: Creation of service account for GCP project "CSCI5410-Assignment4-PartB"

Once the creation of service account "**CSCI5410-Assignment4-PartB-SA**" is completed, we need to create a key pair using the available option "**Add Key**" which generates a JSON file. The content of this JSON file is showed below.

```
{  
  "type": "type",  
  "project_id": "project_id",  
  "private_key_id": "private_key_id",  
  "private_key": "private_key",  
  "client_email": "client_email",  
  "client_id": "client_id",  
  "auth_uri": "auth_uri",  
  "token_uri": "token_uri",  
  "auth_provider_x509_cert_url":  
  "auth_provider_x509_cert_url",  
  "client_x509_cert_url": "client_x509_cert_url"  
}
```

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516

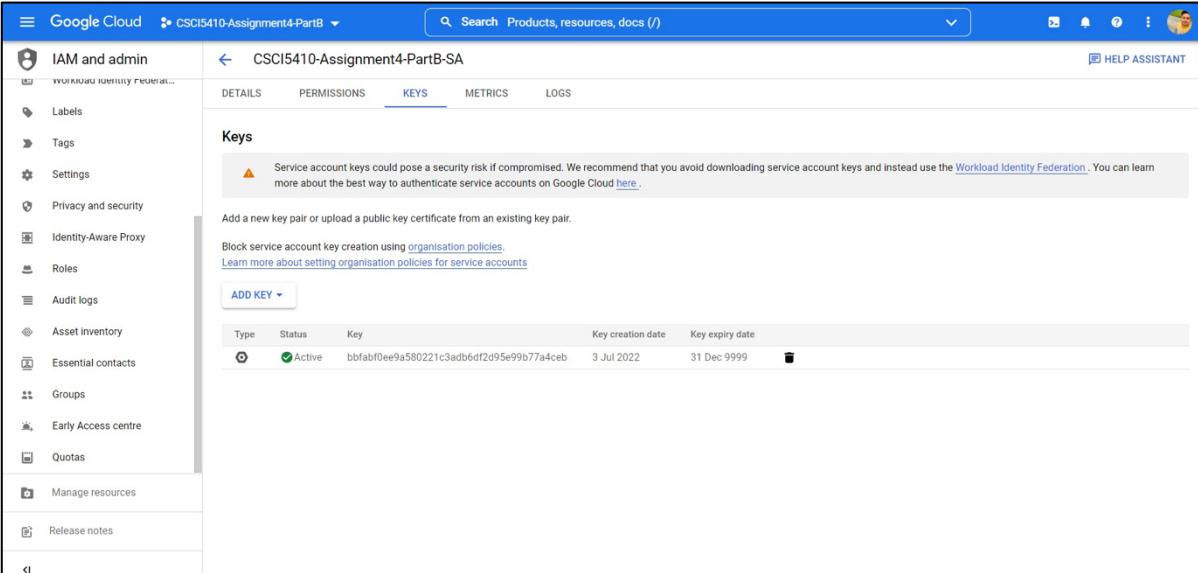
Figure 7 is responsible for displaying the creation of JSON file or key pair inside the “CSCI5410-Assignment4-PartB-SA” service account using the available option “Add Key”.



The screenshot shows the Google Cloud IAM and admin interface for the project "CSCI5410-Assignment4-PartB". The left sidebar lists various service accounts and their details. The main area is focused on the "CSCI5410-Assignment4-PartB-SA" service account, specifically the "Keys" tab. A prominent "ADD KEY" button is located in the center. Below it, there's a table with columns for Type, Status, Key, Key creation date, and Key expiry date. The table currently displays "No rows to display". A note at the top of the Keys section cautions against downloading service account keys and recommends using Workload Identity Federation.

Figure 7: Creation of key Pair for service account “CSCI5410-Assignment4-PartB-SA”

Figure 8 shows the successful creation of key pair in the “CSCI5410-Assignment4-PartB-SA” service account.



This screenshot is identical to Figure 7, but it shows a successfully created key pair. The table now contains one row: a single active key. The key details are: Type (JSON), Status (Active), Key ID (bbfabf0ee9a580221c3adb6df2d95e99b77a4ceb), Key creation date (3 Jul 2022), and Key expiry date (31 Dec 9999). The "ADD KEY" button is no longer visible as the key has been added.

Figure 8: Successful creation of Key Pair for service account “CSCI5410-Assignment4-PartB-SA”

Now we need to enable two APIs in **CSCI5410-Assignment4-PartB** project.

1. Cloud Functions API - cloudfunctions.googleapis.com
2. Google Cloud Build API - cloudbuild.googleapis.com

Figure 9 shows Cloud Function API and **Figure 10** shows enabled Cloud Function API

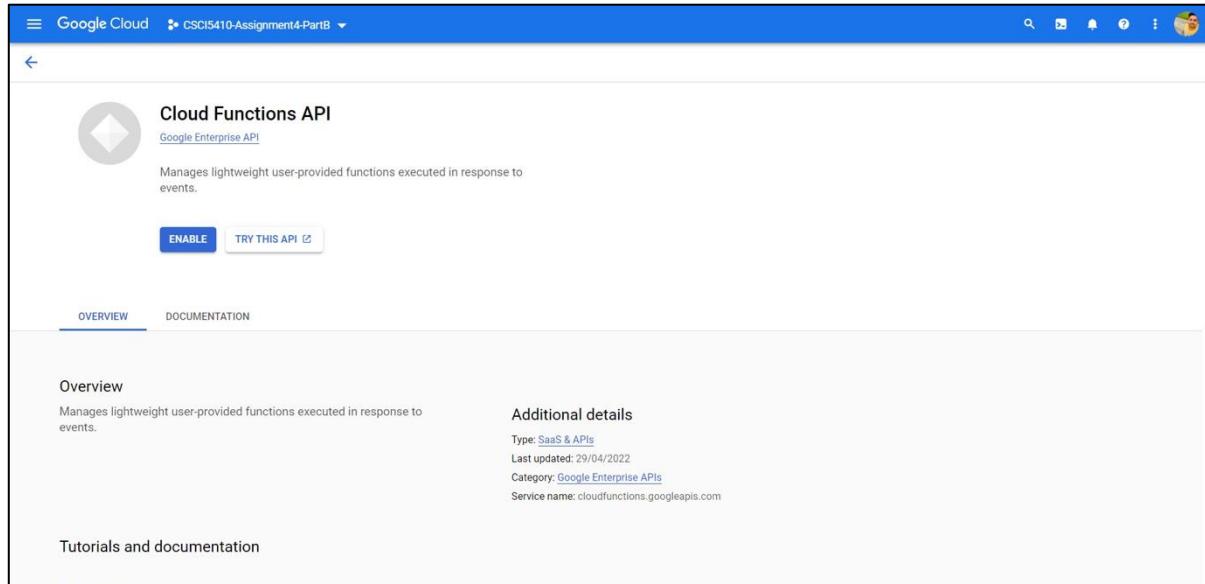


Figure 9: Could Function API

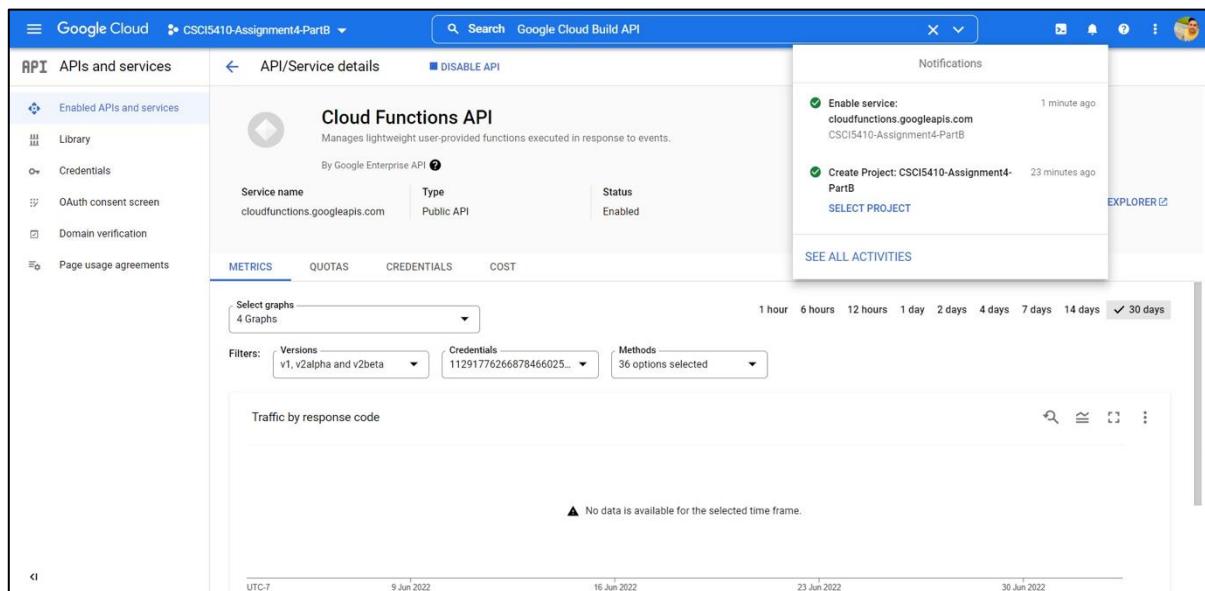


Figure 10: Enabled Cloud Function API

Figure 11 shows Google Cloud build API and Figure 12 Shows the enabled Cloud Build API.

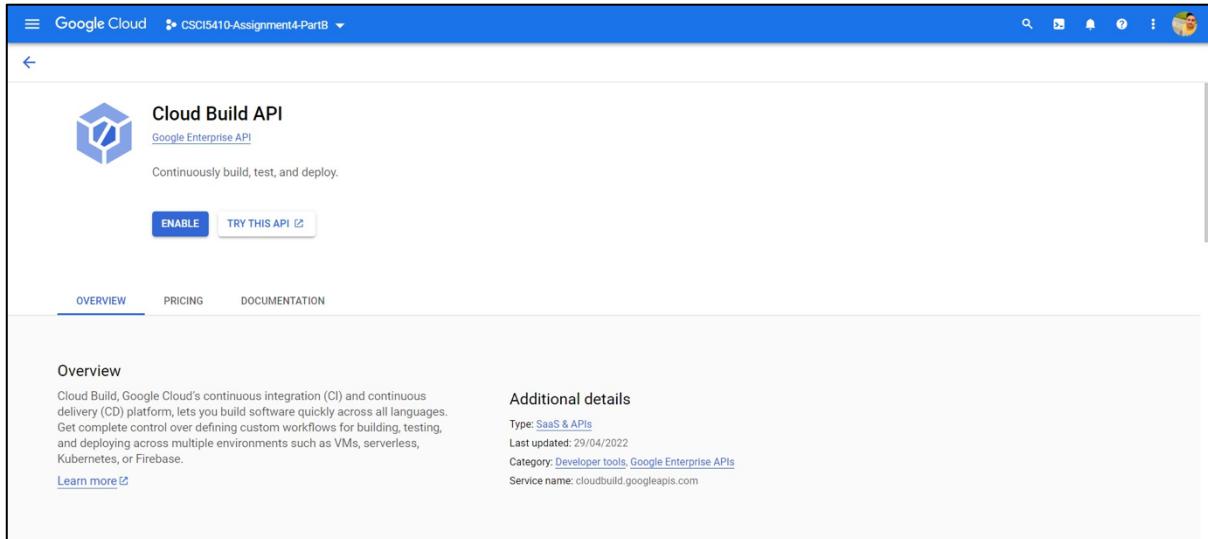


Figure 11: Cloud Build API

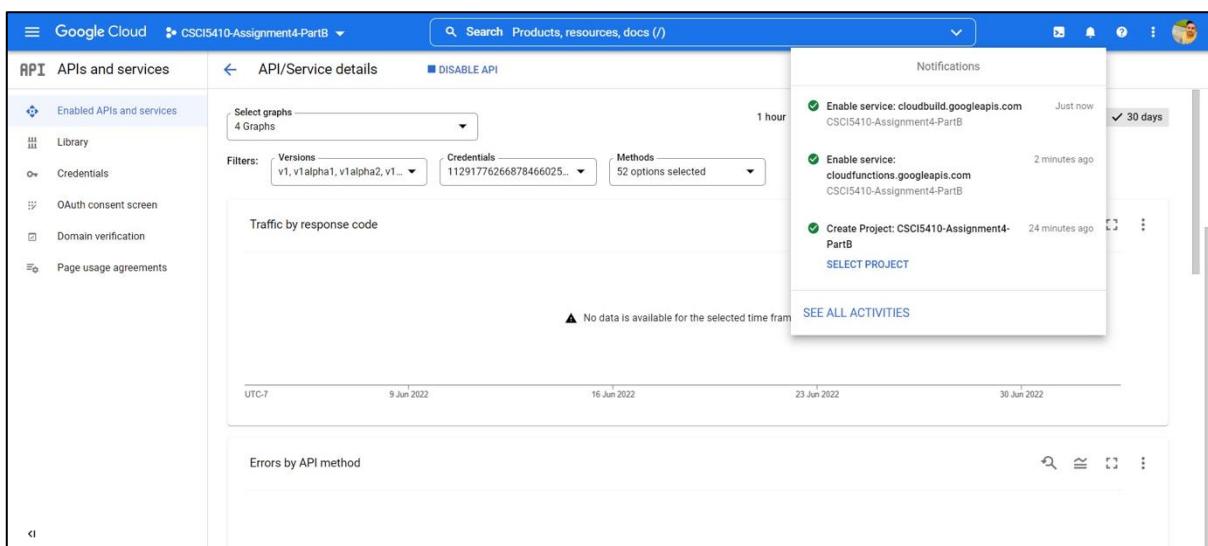


Figure 12: Enabled Cloud Build API

A. Create your 1st storage bucket SourceDataB00xxxxxx and upload the files (from 001 to 299) given in the Train folder. You need to write a script or use the SDK to upload the files on the bucket

Figure 13 shows the empty Cloud Storage with no bucket initially

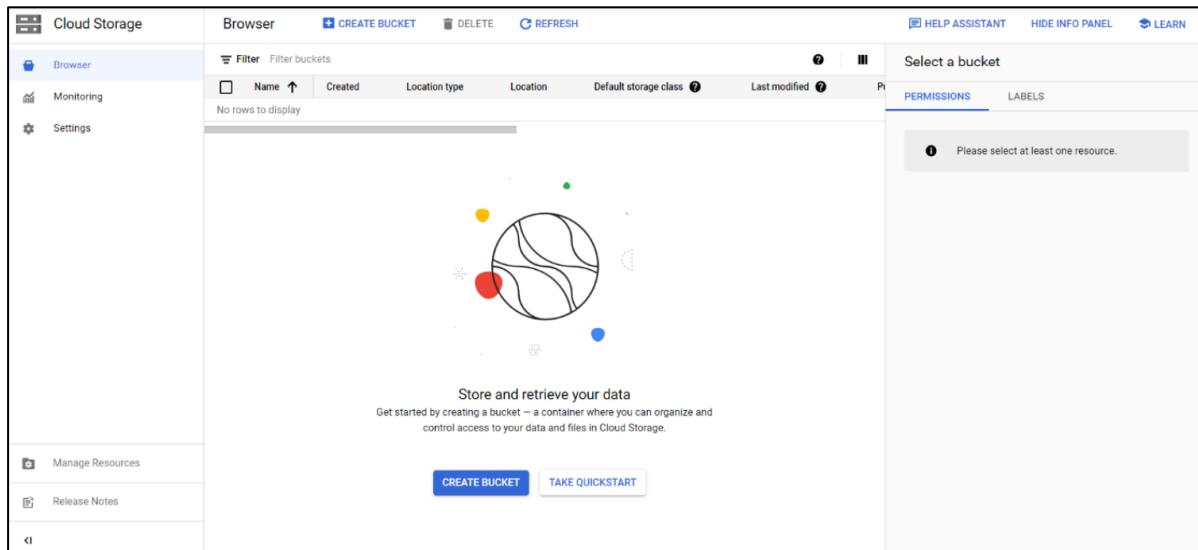
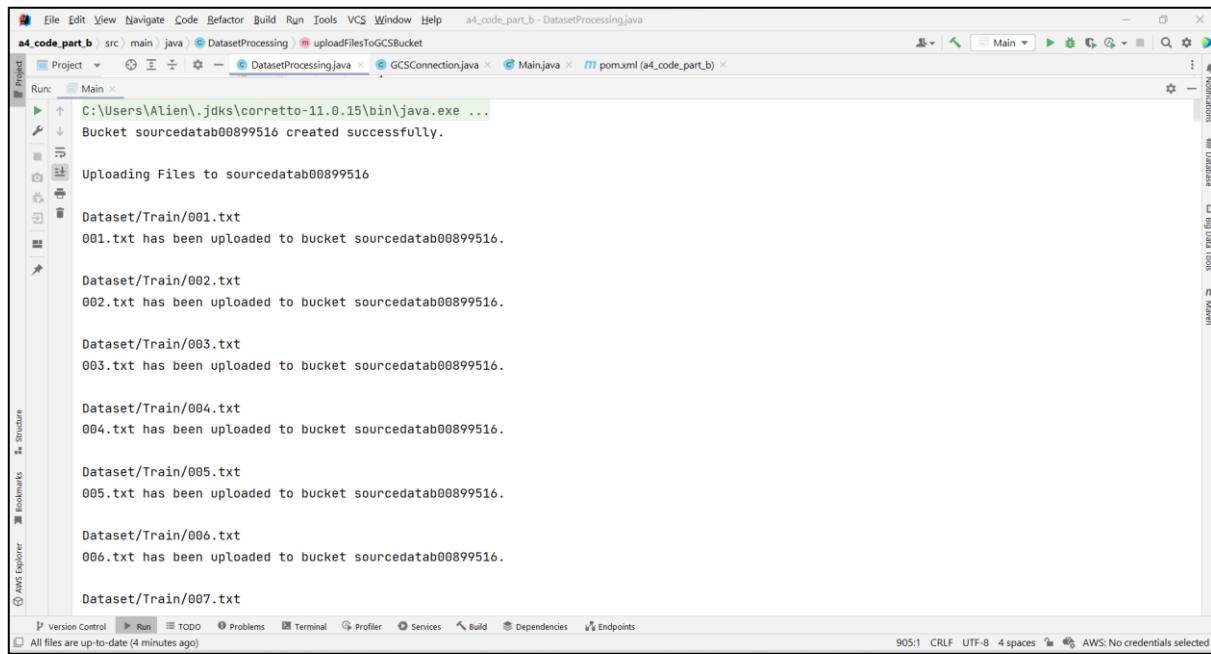


Figure 13: Cloud Function with no bucket

Figure from 13 to 17 shows the execution of program that creates the bucket with name “**sourcedatab00899516**” in google cloud storage. This Program will create this bucket “**sourcedatab00899516**” in when there is no bucket exists with the name “**sourcedatab00899516**”. This program will check the cloud storage before creating the bucket. After creating the bucket, the program will upload all the files inside the **Dataset\Train** folder to the “**sourcedatab00899516**” bucket one by one.

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516



```
C:\Users\Alien\.jdks\corretto-11.0.15\bin\java.exe ...
Bucket sourcedatab00899516 created successfully.

Uploading Files to sourcedatab00899516

Dataset/Train/001.txt
001.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/002.txt
002.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/003.txt
003.txt has been uploaded to bucket sourcedatab00899516.

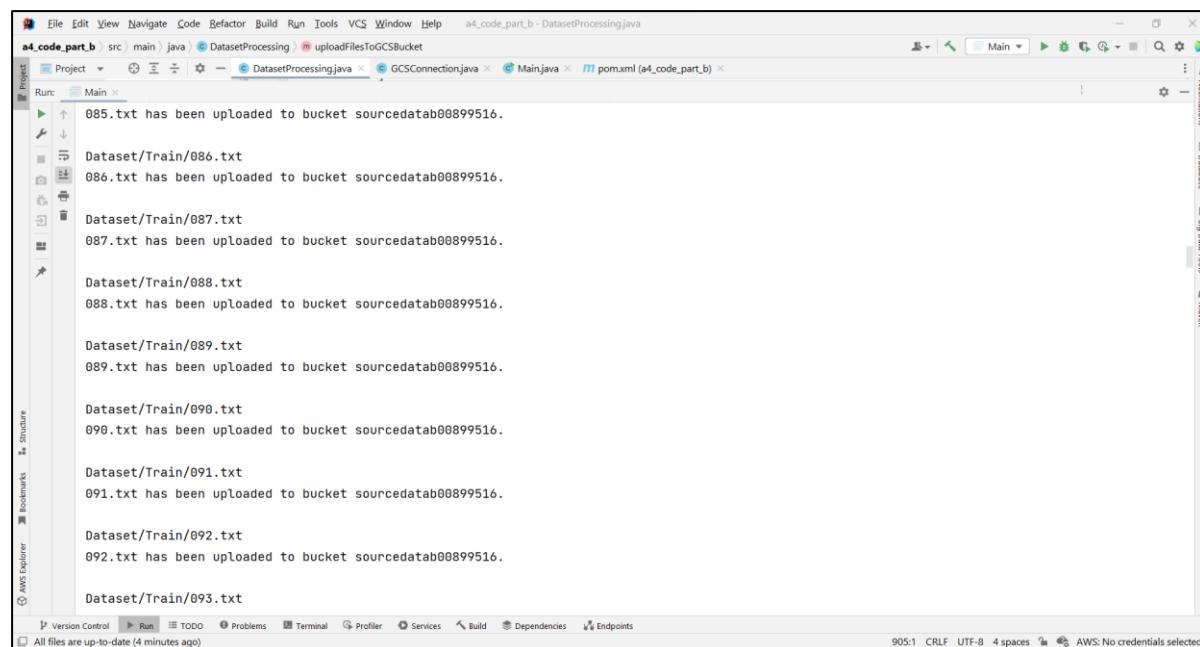
Dataset/Train/004.txt
004.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/005.txt
005.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/006.txt
006.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/007.txt
```

Figure 14: Bucket "sourcedatab00899516" created successfully and then files from train folder uploaded to that bucket



```
085.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/086.txt
086.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/087.txt
087.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/088.txt
088.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/089.txt
089.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/090.txt
090.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/091.txt
091.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/092.txt
092.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/093.txt
```

Figure 15: Bucket "sourcedatab00899516" created successfully and then files from train folder uploaded to that bucket

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516

The screenshot shows an IDE interface with a Java project named "a4_code_part_b". The "Run" tab is selected, displaying the output of the "Main" class. The output window shows the following log entries:

```
128.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/129.txt  
129.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/130.txt  
130.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/131.txt  
131.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/132.txt  
132.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/133.txt  
133.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/134.txt  
134.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/135.txt  
135.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/136.txt
```

The IDE interface includes toolbars, a project tree, and various tabs for version control, run configurations, and AWS services.

Figure 16: Bucket "sourcedatab00899516" created successfully and then files from train folder uploaded to that bucket

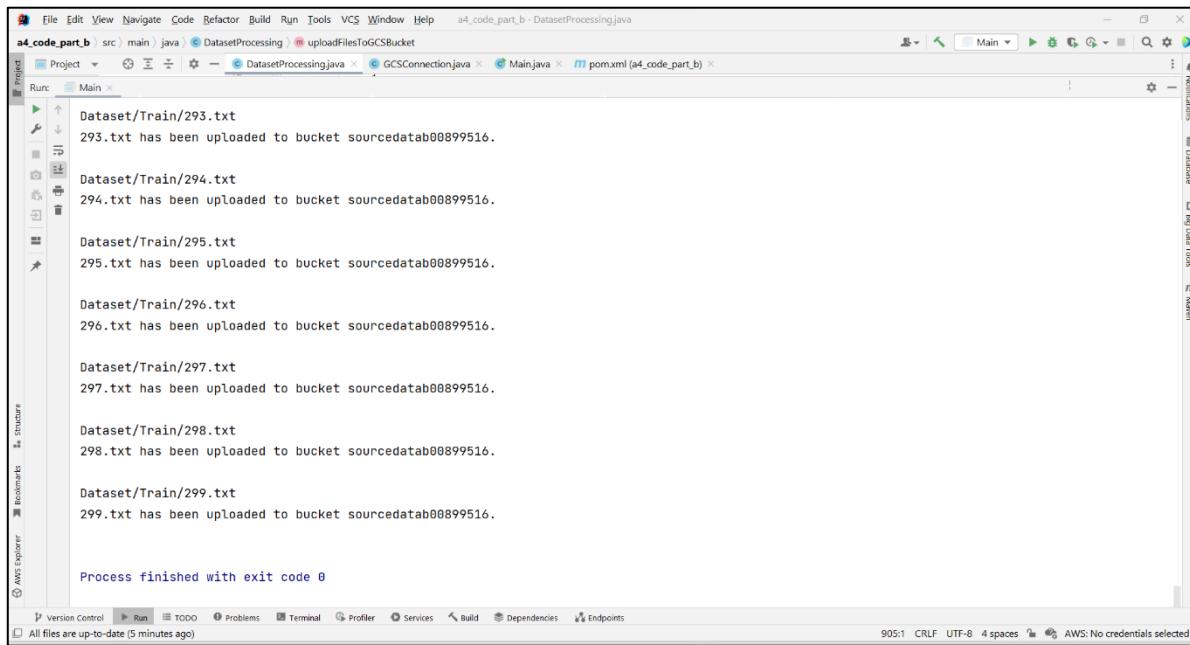
The screenshot shows an IDE interface with a Java project named "a4_code_part_b". The "Run" tab is selected, displaying the output of the "Main" class. The output window shows the following log entries:

```
Dataset/Train/177.txt  
177.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/178.txt  
178.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/179.txt  
179.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/180.txt  
180.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/181.txt  
181.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/182.txt  
182.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/183.txt  
183.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Train/184.txt  
184.txt has been uploaded to bucket sourcedatab00899516.
```

The IDE interface includes toolbars, a project tree, and various tabs for version control, run configurations, and AWS services.

Figure 17: Bucket "sourcedatab00899516" created successfully and then files from train folder uploaded to that bucket

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516



The screenshot shows a Java application running in an IDE. The code is processing files from a 'Train' folder and uploading them to a Google Cloud Storage bucket named 'sourcedatab00899516'. The logs in the terminal window show the following output:

```
Dataset/Train/293.txt
293.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/294.txt
294.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/295.txt
295.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/296.txt
296.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/297.txt
297.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/298.txt
298.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Train/299.txt
299.txt has been uploaded to bucket sourcedatab00899516.

Process finished with exit code 0
```

Figure 18: Bucket "**sourcedatab00899516**" created successfully and then files from train folder uploaded to that bucket

Figure 18 shows the bucket "**sourcedatab00899516**" created successfully on google storage under the project name **CSCI5410-Assignment4-PartB**

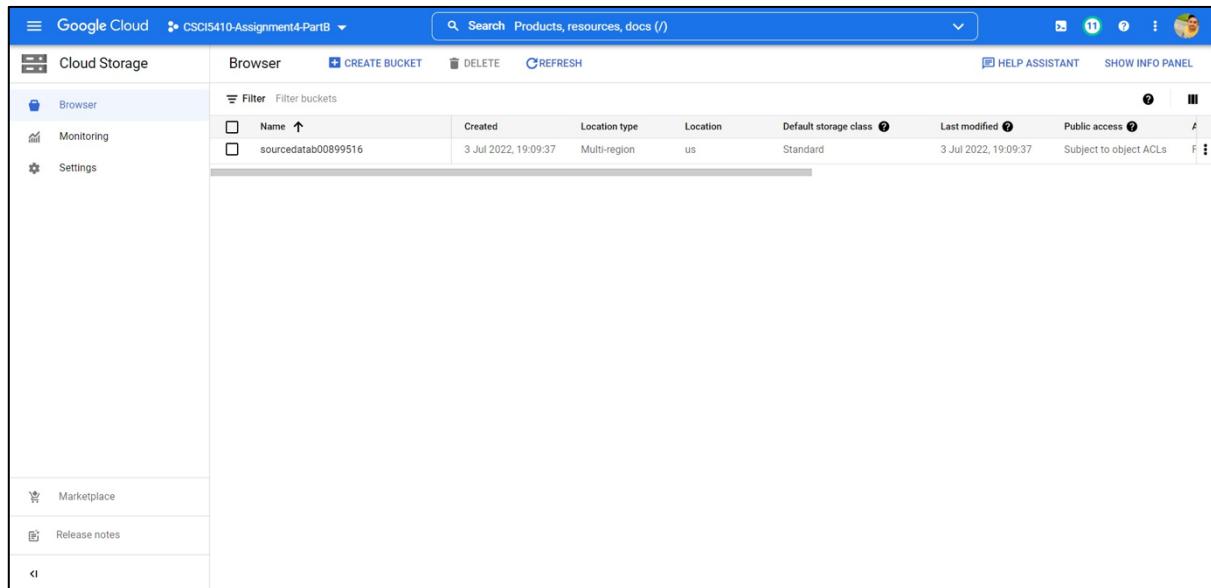


Figure 19: Bucket "**sourcedatab00899516**" created successfully on Google Cloud Storage

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516

Figure 19 to 23 are responsible for showing the uploaded file on “**sourcedatab00899516**” bucket. These files are uploaded from Train folder. There are told 299 files (001.txt to 299.txt) which are uploaded to this bucket.

The screenshot shows the Google Cloud Storage interface for the bucket "sourcedatab00899516". The left sidebar includes links for Cloud Storage, Browser, Monitoring, and Settings. The main area displays the bucket details, showing it was created on 3 Jul 2022, has Standard storage class, and is subject to object ACLs. The "OBJECTS" tab is selected, showing a list of 299 files. The files are named from 001.txt to 299.txt, are all application/octet-stream type, and have a size ranging from 1.7 KB to 4.8 KB. All files were uploaded on 3 Jul 2022. The "Filter by name prefix only" dropdown is set to an empty string. The "Show deleted data" checkbox is unchecked. The "Filter objects and folders" input field is empty.

Figure 20: Files from the train folder uploaded to "sourcedatab00899516" bucket

This screenshot is identical to Figure 20, showing the Google Cloud Storage interface for the same bucket. It displays the same 299 files from the Train folder, all uploaded on 3 Jul 2022. The list includes files like 105.txt, 106.txt, 107.txt, etc., up to 299.txt. The interface elements, including the sidebar, bucket details, and file list, are identical.

Figure 21: Files from the train folder uploaded to "sourcedatab00899516" bucket

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516

The screenshot shows the Google Cloud Storage interface. On the left, there's a sidebar with options like 'Cloud Storage', 'Browser', 'Monitoring', 'Settings', 'Marketplace', and 'Release notes'. The main area is titled 'Bucket details' and shows a list of files under a folder named 'train'. The table has columns for file name, size, content type, modified date, storage class, and access control. All files are 'application/octet-stream' type, modified on 3 Jul 2022, and have 'Standard' storage class. They are all set to 'Not public'. The list contains 299 entries, numbered from 282.txt to 299.txt. The bottom right of the table shows 'Rows per page: 50' and '251 - 299 of 299'.

File	Size	Type	Modified	Storage Class	Access Control	Owner
282.txt	3.3 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
283.txt	4.2 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
284.txt	975 B	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
285.txt	4.3 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
286.txt	4.8 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
287.txt	1.9 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
288.txt	4.6 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
289.txt	2.4 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
290.txt	1.9 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
291.txt	4.1 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
292.txt	2.4 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
293.txt	4 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
294.txt	4.5 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
295.txt	3.7 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
296.txt	3.8 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
297.txt	2.4 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
298.txt	2.5 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public
299.txt	4.2 KB	application/octet-stream	3 Jul 2022	Standard	3 Jul 2022	Not public

Figure 22: Files from the train folder uploaded to "sourcedatab00899516" bucket

B. Once a file is uploaded, a cloud function - “generateVector” should extract words from all the files (remove the stop words). Then compute Levenshtein distance between the Current, and Next word.

Figure 22 is responsible for representing the creation of cloud function with name **generateVector** and the memory allocated to this cloud function is 512 MB and the timeout is set to 540 seconds. With an event type Finalize/Create, a trigger is created and set it to the cloud storage bucket **“sourcedatab00899516”**. So, whenever a new file is uploaded to the **“sourcedatab00899516”** bucket, the **generateVector** cloud function will be executed.

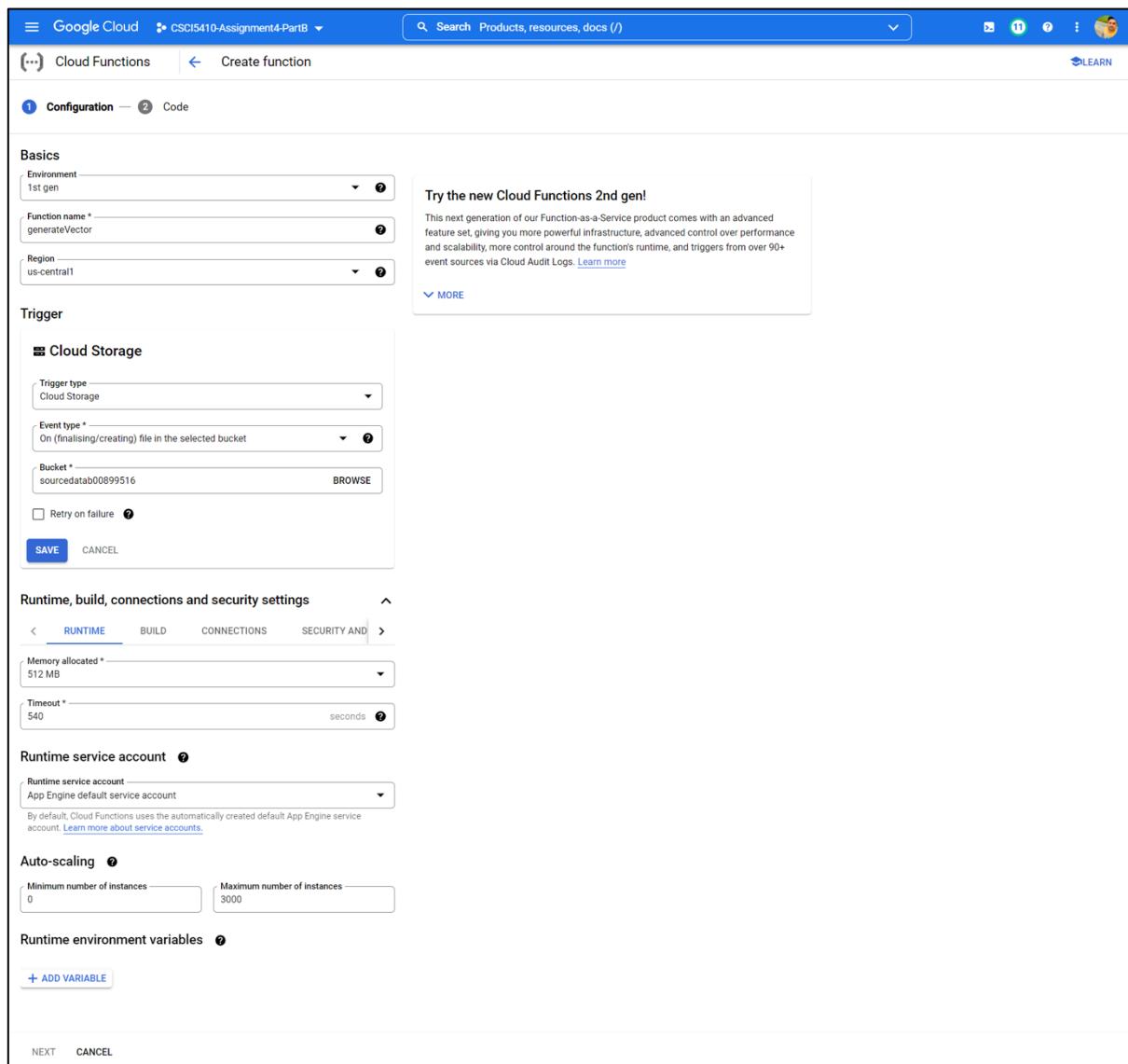
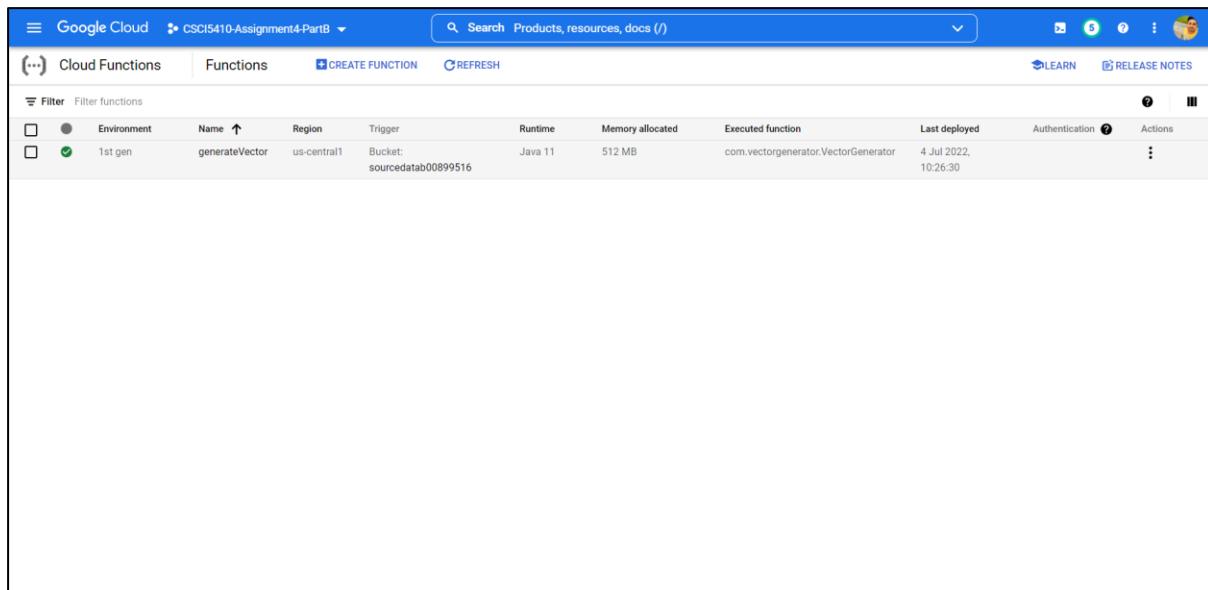


Figure 23: Creation of **generateVector** cloud function

Figure 24 is responsible for representing the successful deployment of cloud function with name “generateVector”



Figure

24: Successfully deployed **generateVector** cloud function

Figure 25 is responsible for showing the code of “generateVector”. The code for the “generateVector” is directly written inside the “generateVector” cloud function.

A screenshot of the "Edit function" page for the "generateVector" function. The top navigation bar shows "Cloud Functions" and "Edit function". The left sidebar has "Configuration" selected. The main area shows the "Code" tab. The "Runtime" is set to "Java 11". The "Entry point" is "com.vectorgenerator.VectorGenerator". The "Source code" section shows an "Inline Editor" with the following Java code:

```
1 package com.vectorgenerator;
2 import com.google.cloud.functions.BackgroundFunction;
3 import com.google.cloud.functions.Context;
4 import com.google.cloud.storage.BlobId;
5 import com.google.cloud.storage.BlobInfo;
6 import com.google.cloud.storage.BucketInfo;
7 import com.google.cloud.storage.Storage;
8 import com.google.cloud.storage.StorageOptions;
9
10 import java.util.logging.Logger;
11
12 import com.google.cloud.storage.Blob;
13 import com.google.api.gax.paging.Page;
14
15 import java.util.ArrayList;
16 import java.util.Arrays;
17 import java.util.List;
18 import java.util.stream.Collectors;
19 import java.util.stream.Stream;
20
21
22 public final class VectorGenerator implements BackgroundFunction<VectorGenerator> {
23     private static final Logger LOGGER = Logger.getLogger(VectorGenerator.class.getName());
24     private static final List<String> STOP_WORDS_LIST = Arrays.asList("i", "me", "my",
25         "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself",
26         "yourselves");
27 }
```

The bottom of the screen shows buttons for "PREVIOUS", "DEPLOY", and "CANCEL".

Figure 25: Added Code for “generateVector” Cloud Function

Figure 26 and 27 shows the logs which are generated when “**generateVector**” cloud function is triggered. This cloud function will be triggered when files from Train folder were uploaded to the “**sourcedatab00899516**”. And when this function is triggered, it will also generate logs. This log provides lots of information like the files from the “**sourcedatab00899516**” bucket will be processed one by one. Then it will also show the creation of new bucket “**traindataab00899516**’ where **trainVectors.csv** will be uploaded.

The screenshot shows the Google Cloud Logging interface. The left sidebar lists 'Operations' and 'Logging'. Under 'Logs Explorer', the 'Query' tab is selected, showing 'Recent (5)' log entries. The logs are listed in a table with columns: SEVERITY, TIMESTAMP, PLOT, SUMMARY, and EDIT. The logs are timestamped from July 07, 2022, at 10:30:33 PDT to 10:38:33 PDT. Most logs have a severity of 'INFO' and are labeled with 'generateVector'. The 'PLOT' column shows small icons representing the log type. The 'SUMMARY' column contains detailed log messages, such as file names like '280.txt', '283.txt', and '284.txt', and descriptions of the function's operation, like 'List of words after removing stop words.' and 'blob name: 268.txt'.

Figure 26: Generated logs messages on successful execution of cloud function “generatedVector”

This screenshot is identical to Figure 26, showing the same Google Cloud Logging interface and log entries for the 'generatedVector' function. It displays the same timestamp range, log types, and detailed log messages about file processing and bucket creation.

Figure 27: Generated logs messages on successful execution of cloud function “generatedVector”

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516

Figure 28 represents the newly created bucket named as “**traindatab00899516**”

The screenshot shows the Google Cloud Storage Browser interface. The left sidebar has 'Cloud Storage' selected. The main area displays a table of buckets. The newly created bucket, 'traindatab00899516', is listed with the following details:

Name	Created	Location type	Location	Default storage class	Last modified	Public access
gcf-sources-96811373779-us-central1	3 Jul 2022, 20:53:58	Region	us-central1	Standard	3 Jul 2022, 20:53:58	Not public
sourcedatab00899516	3 Jul 2022, 19:09:37	Multi-region	us	Standard	4 Jul 2022, 10:26:24	Subject to object ACLs
traindatab00899516	4 Jul 2022, 10:36:52	Multi-region	us	Standard	4 Jul 2022, 10:36:52	Subject to object ACLs
us.artifacts.csc5410-assignment4-pa...	3 Jul 2022, 20:55:24	Multi-region	us	Standard	3 Jul 2022, 20:55:24	Subject to object ACLs

Figure 28: Successfully created new bucket "**traindatab00899516**"

C. This file “trainVector.csv” is saved in a new bucket TrainDataB00xxxxxx.

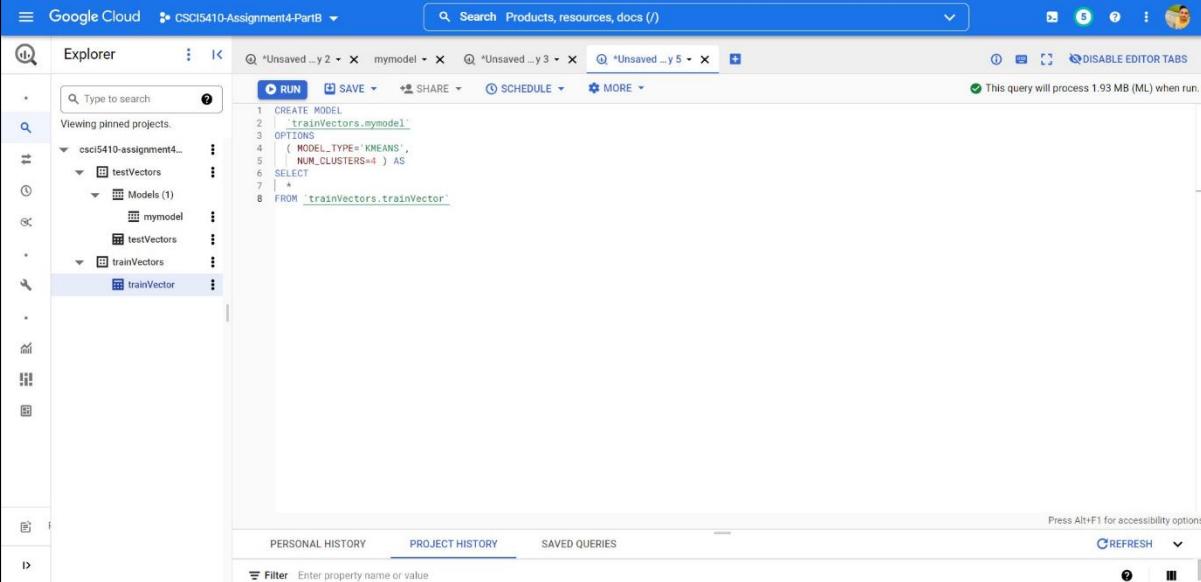
Figure 29 shows the **trainVector.csv** file created and uploaded to “**traindatab00899516**” bucket.

Figure 29: Successfully created "trainVectors.csv" file in "traindatab00899516" bucket

Figure 30 shows the content of **trainVector.csv** file, which is downloaded from “**traindatab00899516**” Bucket. This **trainVector.csv** file has three columns. The name of this columns is current word, the next word and Levenshtein distance.

Figure 30: Content of trainVector.csv

D. GCP ML should get the training data for a clustering algorithm (KMeans) from the TrainDataB00xxxxxx bucket.



The screenshot shows the Google Cloud ML Engine interface. In the top navigation bar, it says "Google Cloud" and "CSCI5410-Assignment4-PartB". The main area is titled "Explorer" and shows a tree structure of projects and models. A query editor window is open with the following SQL-like code:

```
1 CREATE MODEL
2   trainVectors.mymodel
3   OPTIONS
4     ( MODEL_TYPE='KMEANS',
5       NUM_CLUSTERS=4 ) AS
6   SELECT
7     *
8   FROM trainVectors.trainVector;
```

Below the code, there are tabs for "PERSONAL HISTORY", "PROJECT HISTORY", and "SAVED QUERIES". At the bottom, there is a search bar with the placeholder "Filter Enter property name or value".

Figure 31: Training data generation for the KMeans algorithm from the TrainDataB00xxxxxx bucket using GCP ML

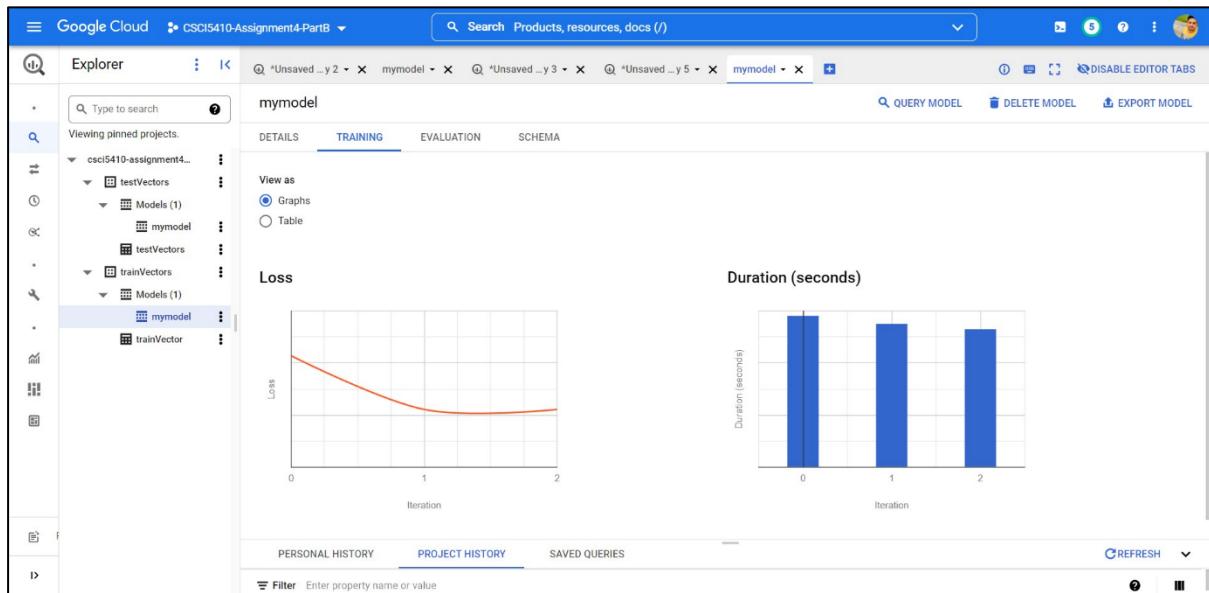


Figure 32: Training data generation for the KMeans algorithm from the TrainDataB00xxxxxx bucket using GCP ML

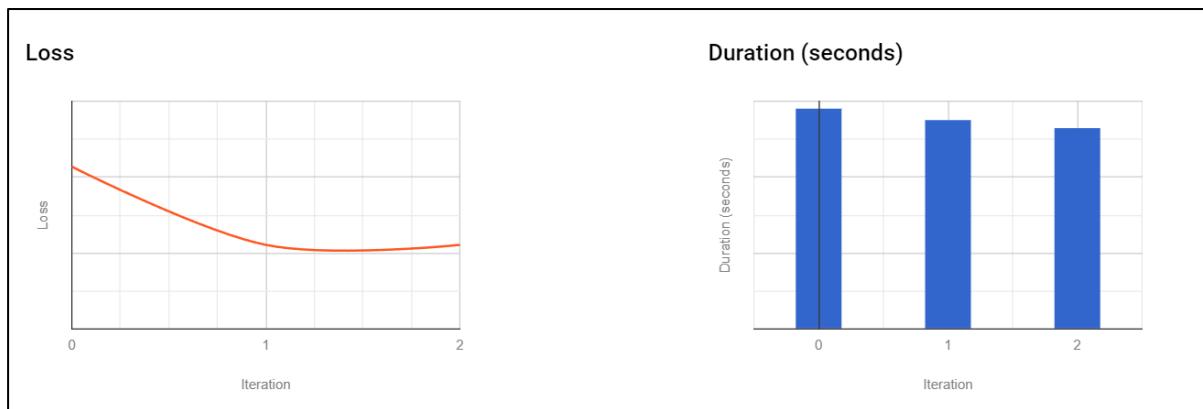


Figure 33: Training data generation for the KMeans algorithm from the TrainDataB00xxxxx bucket using GCP ML

E. Once the training is done, like point (a), upload the test files given in the Test folder (300 to 401) to SourceDataB00xxxxxx.

Figure 34 to 36 is responsible for showing the files that are being uploaded from test folder to “sourcedatab00899516”. This program does not create new bucket instead it will show the message like “sourcedatab00899516” bucket already exists and utilize that bucket to upload all files from Dataset\Test\ folder. Total 100 files will be uploaded (300.txt to 401.txt) to “sourcedatab00899516”

```

C:\Users\Alien\.jdks\corretto-11.0.15\bin\java.exe ...
Bucket sourcedatab00899516 exists already.

Uploading Files to sourcedatab00899516

Dataset/Test/300.txt
300.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Test/301.txt
301.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Test/302.txt
302.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Test/303.txt
303.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Test/304.txt
304.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Test/305.txt
305.txt has been uploaded to bucket sourcedatab00899516.

Dataset/Test/306.txt

```

Figure 34: Uploading Dataset\Test\ folder files to "sourcedatab00899516"

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516

The screenshot shows the AWS Lambda IDE interface. The project is named 'a4_code_part_b'. The 'Run' tab is selected, showing the output of the 'Main' function. The console output displays the following messages:

```
326.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/327.txt  
327.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/328.txt  
328.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/329.txt  
329.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/330.txt  
330.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/331.txt  
331.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/332.txt  
332.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/333.txt  
333.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/334.txt
```

The bottom status bar indicates: Build completed successfully in 5 sec. 553 ms (31 minutes ago). 314:1 CRLF UTF-8 4 spaces AWS: No credentials selected.

Figure 35: Uploading Dataset\Test\ folder files to "sourcedatab00899516"

The screenshot shows the AWS Lambda IDE interface. The project is named 'a4_code_part_b'. The 'Run' tab is selected, showing the output of the 'Main' function. The console output displays the following messages:

```
Dataset/Test/395.txt  
395.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/396.txt  
396.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/397.txt  
397.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/398.txt  
398.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/399.txt  
399.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/400.txt  
400.txt has been uploaded to bucket sourcedatab00899516.  
Dataset/Test/401.txt  
401.txt has been uploaded to bucket sourcedatab00899516.
```

At the end of the output, it says: Process finished with exit code 0.

The bottom status bar indicates: Build completed successfully in 5 sec. 553 ms (32 minutes ago). 314:1 CRLF UTF-8 4 spaces AWS: No credentials selected.

Figure 36: Uploading Dataset\Test\ folder files to "sourcedatab00899516"

CSCI 5410: Assignment #4 (Part B) | Meet Patel B00899516

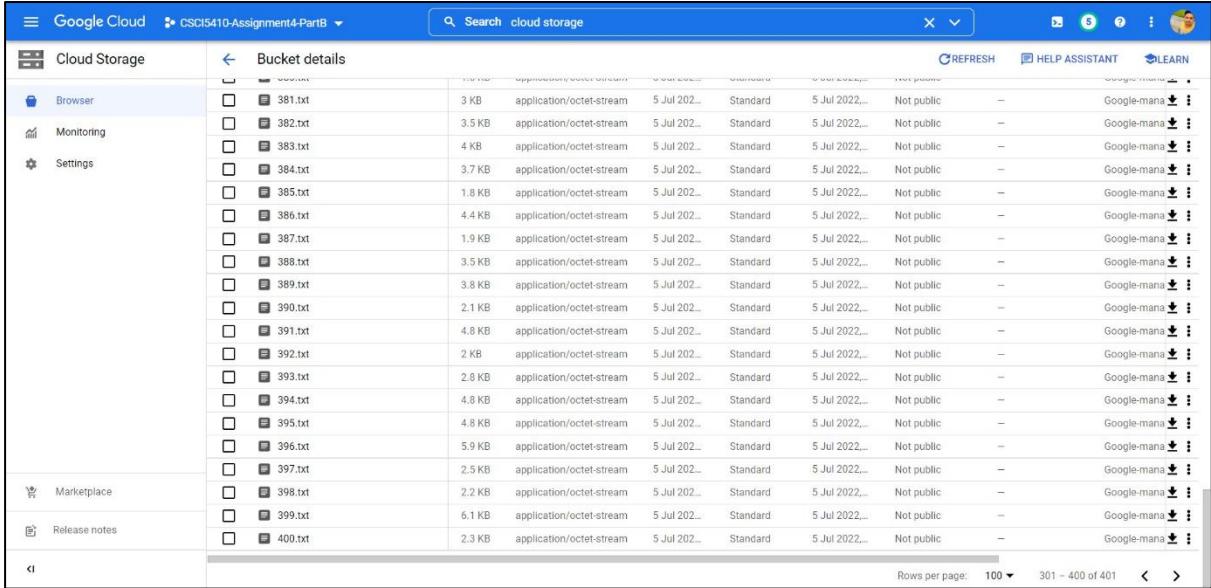
Figure 34 to 36 shows the test files that are being successfully uploaded to **sourcedatab00899516** bucket on google storage. Total 102 files were uploaded (301.txt to 400.txt) on google storage.

The screenshot shows the Google Cloud Storage interface. The left sidebar has 'Cloud Storage' selected. The main area shows 'Bucket details' for 'sourcedatab00899516'. The bucket is located in 'us (multiple regions in United States)', has 'Standard' storage class, 'Public access' set to 'Subject to object ACLs', and 'Protection' set to 'None'. Below this are tabs for 'OBJECTS', 'CONFIGURATION', 'PERMISSION', 'PROTECTION', and 'LIFECYCLE'. The 'OBJECTS' tab is active, showing a list of objects. The list includes objects from 301.txt to 311.txt, each with details like size (e.g., 5.8 KB), type (application/octet-stream), creation date (5 Jul 2022), storage class (Standard), last modified date (5 Jul 2022), public access (Not public), version history (none), and encryption status (Google-managed). There are also buttons for 'UPLOAD FILES', 'UPLOAD FOLDER', 'CREATE FOLDER', 'MANAGE HOLDS', 'DOWNLOAD', and 'DELETE'.

Figure 37: Dataset/Text folder files uploaded to "sourcedatab00899516" bucket on google cloud storage

This screenshot is identical to Figure 37, showing the Google Cloud Storage interface for the 'sourcedatab00899516' bucket. It displays the same list of objects from 344.txt to 365.txt, all with 'Not public' public access and 'Google-managed' encryption. The interface includes the same navigation and management tools as Figure 37.

Figure 38: Dataset/Text folder files uploaded to "sourcedatab00899516" bucket on google cloud storage



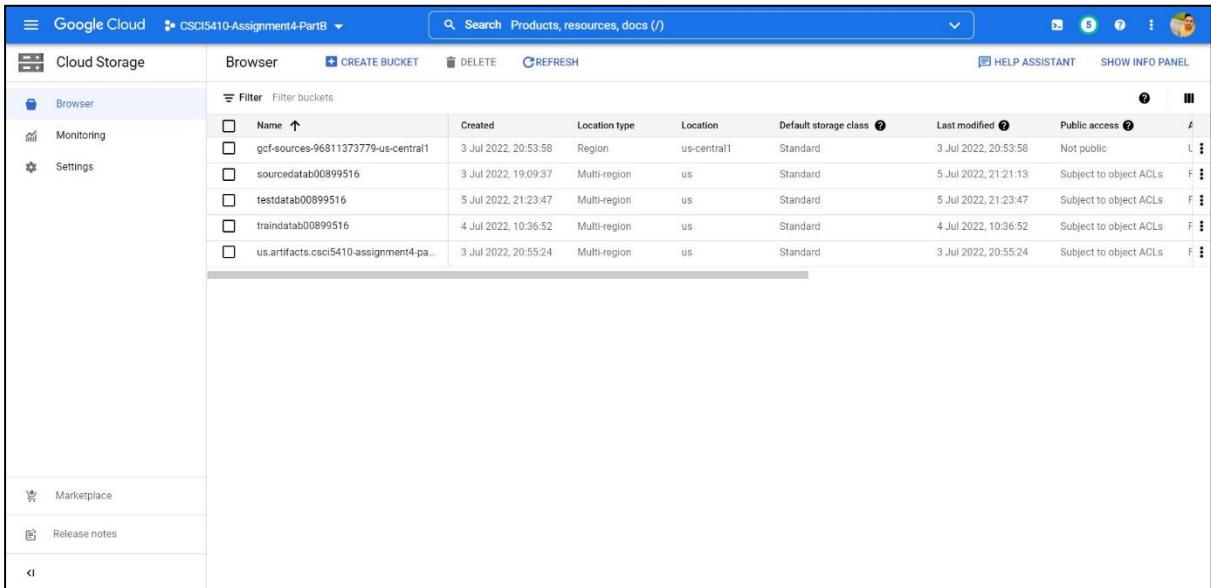
The screenshot shows the Google Cloud Storage interface with the project 'CSCI5410-Assignment4-PartB'. The left sidebar has sections for 'Cloud Storage', 'Browser', 'Monitoring', 'Settings', 'Marketplace', and 'Release notes'. The main area is titled 'Bucket details' and shows a list of files under the 'sourcedatab00899516' bucket. The files are listed in descending order by name:

Name	Size	Type	Created	Last modified	Storage class	Public access	Action
381.txt	3 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
382.txt	3.5 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
383.txt	4 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
384.txt	3.7 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
385.txt	1.8 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
386.txt	4.4 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
387.txt	1.9 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
388.txt	3.5 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
389.txt	3.8 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
390.txt	2.1 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
391.txt	4.8 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
392.txt	2 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
393.txt	2.8 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
394.txt	4.8 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
395.txt	4.8 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
396.txt	5.9 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
397.txt	2.5 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
398.txt	2.2 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
399.txt	6.1 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮
400.txt	2.3 KB	application/octet-stream	5 Jul 202...	5 Jul 202...	Standard	Not public	⋮

Figure 39: Dataset/Text folder files uploaded to "sourcedatab00899516" bucket on google cloud storage

F. The cloud function generateVector computes the distance vector same as point (b) and store it in “testVector.csv”

Figure 37 is responsible for displaying the newly created bucket with name **testdatab00899516**



The screenshot shows the Google Cloud Storage interface with the project 'CSCI5410-Assignment4-PartB'. The left sidebar has sections for 'Cloud Storage', 'Browser', 'Monitoring', 'Settings', 'Marketplace', and 'Release notes'. The main area shows a list of buckets:

Name	Created	Location type	Location	Default storage class	Last modified	Public access
gcf-sources-96811373779-us-central1	3 Jul 2022, 20:53:58	Region	us-central1	Standard	3 Jul 2022, 20:53:58	Not public
sourcedatab00899516	3 Jul 2022, 19:09:37	Multi-region	us	Standard	5 Jul 2022, 21:21:13	Subject to object ACLs
testdatab00899516	5 Jul 2022, 21:23:47	Multi-region	us	Standard	5 Jul 2022, 21:23:47	Subject to object ACLs
trainddb00899516	4 Jul 2022, 10:36:52	Multi-region	us	Standard	4 Jul 2022, 10:36:52	Subject to object ACLs
us.artifacts.csci5410-assignment4-pa...	3 Jul 2022, 20:55:24	Multi-region	us	Standard	3 Jul 2022, 20:55:24	Subject to object ACLs

Figure 40: Newly created bucket with bucket name **testdatab00899516**

Figure 38 is responsible for showing the testVector.csv which is uploaded to the newly created bucket **testdatab00899516**

The screenshot shows the Google Cloud Storage interface. On the left, there's a sidebar with 'Cloud Storage' selected, followed by 'Browser', 'Monitoring', and 'Settings'. Below this is a 'Marketplace' section. The main area is titled 'testdataab00899516' and shows bucket details: Location (us), Storage class (Standard), Public access (Subject to object ACLs), and Protection (None). There are tabs for 'OBJECTS', 'CONFIGURATION', 'PERMISSION', 'PROTECTION', and 'LIFECYCLE'. Under 'OBJECTS', it says 'Buckets > testdataab00899516'. Below that are buttons for 'UPLOAD FILES', 'UPLOAD FOLDER', 'CREATE FOLDER', 'MANAGE HOLDS', 'DOWNLOAD', and 'DELETE'. A search bar says 'Filter objects and folders'. A table lists one object: 'testVector.csv' (1.8 MB, application/octet-stream, created 5 Jul 2022, last modified 5 Jul 2022, public access Not public, version history -, encryption Google-managed).

Figure 41: **textVector.csv** file uploaded to newly created bucket **testdataab00899516** in cloud storage

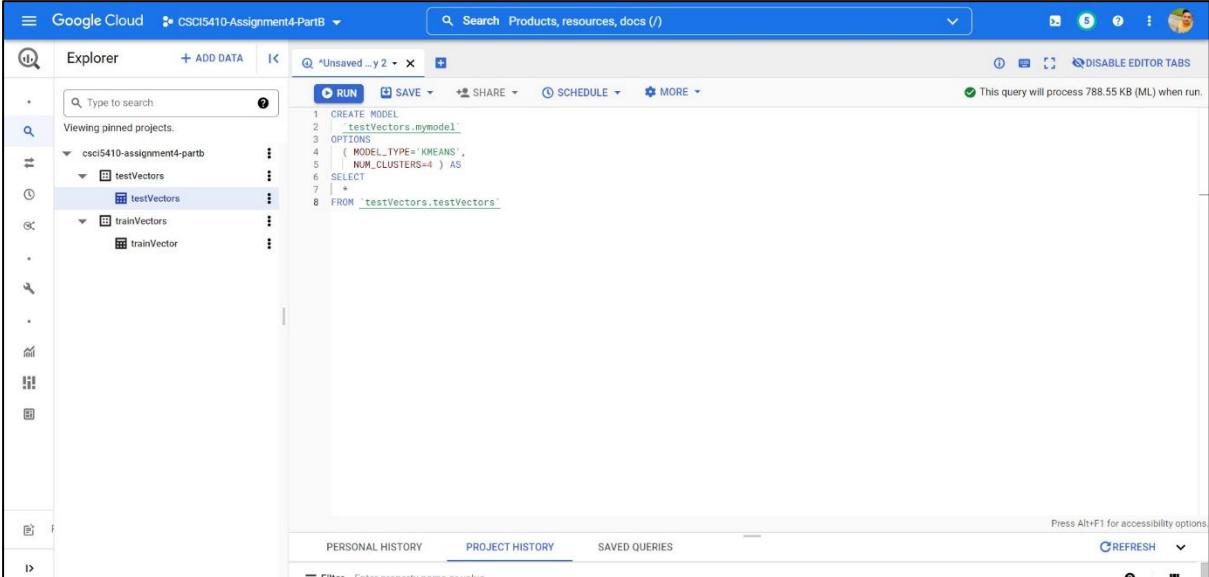
G. This file (**testVector.csv**) is saved in another new bucket, **TestDataB00xxxxxx**.

Figure 39 shows the content of **testVector.csv** file, which is downloaded from “**testdataab00899516**” Bucket. This **testVector.csv** file has three columns. The name of this columns is current word, the next word and Levenshtein distance.

The screenshot shows an Excel spreadsheet with the title 'testVectors'. The data is organized into three columns: 'Current_Word' (A), 'Next_Word' (B), and 'Levenshtein_distance' (C). The first row contains column headers. The data starts at row 2 and continues through row 28. The 'Current_Word' column lists words like 'apple', 'mac', 'mini', 'gets', 'warm', 'welcome', 'mac', 'mini', 'welcomed', 'apple', 'fans', 'industry', 'experts', 'pc', 'users', 'release', 'tiny', 'low-cost', 'machine', 'seen', 'good', 'move', 'apple', 'currently', 'small', 'share', 'desktop', and 'computer'. The 'Next_Word' column lists words like 'mac', 'mini', 'gets', 'warm', 'welcome', 'mac', 'mini', 'welcomed', 'apple', 'fans', 'industry', 'experts', 'pc', 'users', 'release', 'tiny', 'low-cost', 'machine', 'seen', 'good', 'move', 'apple', 'currently', 'small', 'share', 'desktop', and 'computer'. The 'Levenshtein_distance' column lists numerical values ranging from 3 to 8.

Figure 42: **testVector.csv** file content

H. GCP ML should get the test data for the KMeans algorithm from the TestDataB00xxxxxx bucket.



The screenshot shows the Google Cloud BigQuery interface. The left sidebar displays a project structure under 'csci5410-assignment4-partB'. In the main area, a query is being run:

```
1 CREATE MODEL
2   testVectors.mymodel
3   OPTIONS(
4     MODEL_TYPE='KMEANS',
5     NUM_CLUSTERS=4 ) AS
6 SELECT
7   *
8 FROM `testVectors.testVectors`
```

The status bar at the bottom right indicates: "This query will process 788.55 KB (ML) when run."

Figure 43: Test data generation for the KMeans algorithm from the TestDataB00xxxxxx bucket using GCP ML

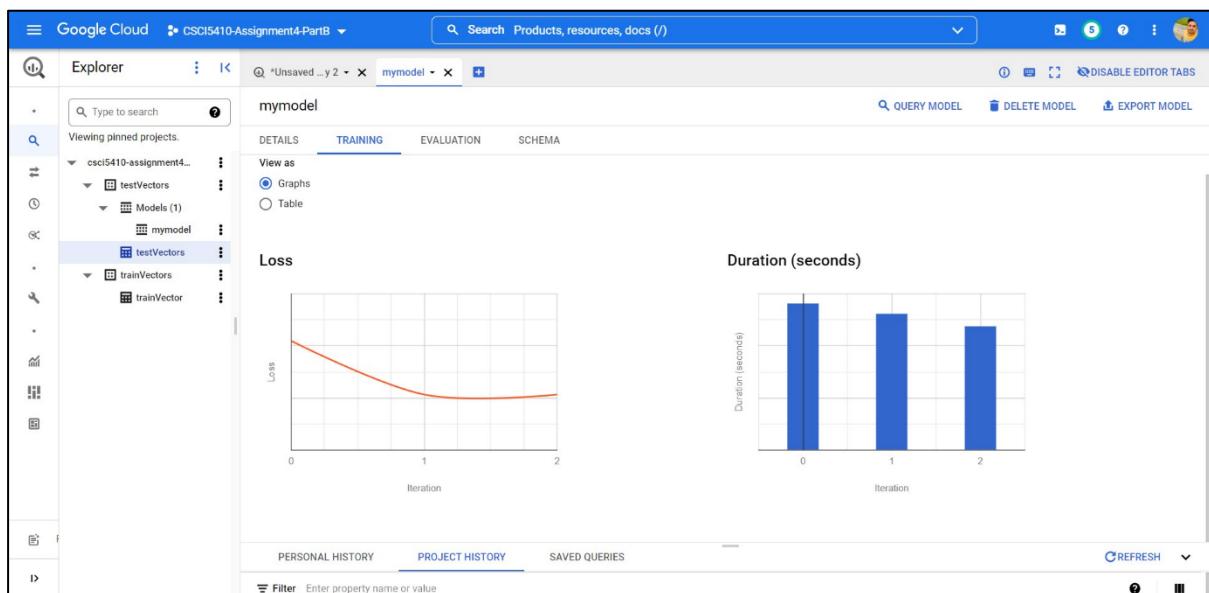


Figure 44: Test data generation for the KMeans algorithm from the TestDataB00xxxxxx bucket using GCP ML

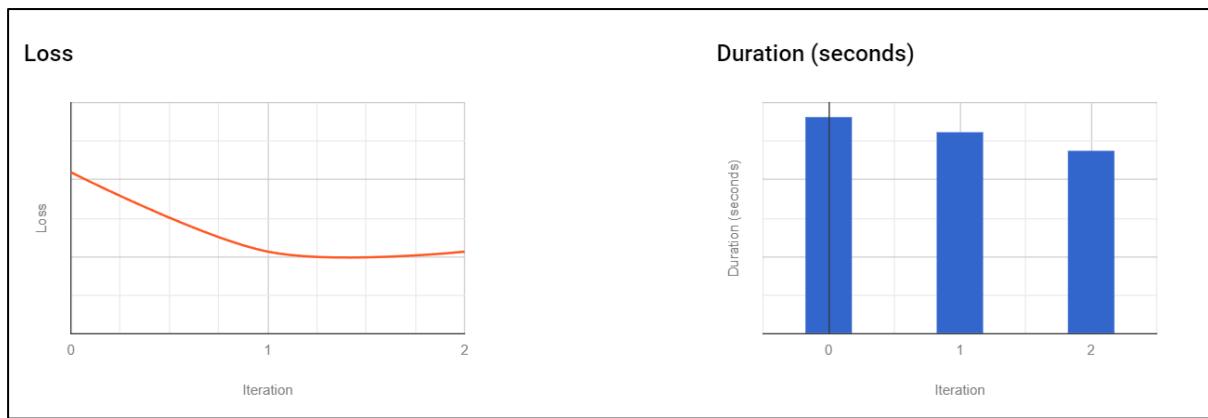


Figure 45: Test data for the KMeans algorithm from the TestDataB00xxxxxx bucket

- I. Finally, write a code or configure a service to obtain information about clusters (e.g. centroids, cluster numbers, outliers etc. which are generated by GCP ML), and display the clusters.

Information about cluster for [testvectors.csv](#) generated by GCP ML

A screenshot of the Google Cloud BigQuery interface. The left sidebar shows a project structure with a pinned project 'csci5410-assignment4...' containing 'testVectors' and 'trainVectors' datasets, and a 'Models (1)' section with a single model named 'mymodel'. The main area displays a SQL query:

```
1 SELECT
2   *
3   FROM
4     ML.PREDICT(MODEL `testVectors.mymodel`,
5       (
6         SELECT
7           Current_Word,
8           Next_Word,
9           Levenshtein_distance
10          FROM
11            `testVectors.testVectors`))
```

The query is highlighted in blue. The top navigation bar shows the project name 'CSCI5410-Assignment4-PartB' and the current tab 'RUN'. There are also tabs for 'SAVE', 'SHARE', 'SCHEDULE', and 'MORE'. A note indicates that the query will process 1.52 MB when run. The bottom navigation bar includes 'PERSONAL HISTORY', 'PROJECT HISTORY', 'SAVED QUERIES', and a 'REFRESH' button.

Figure 46: Information about clusters for testvector.csv generated by GCP ML

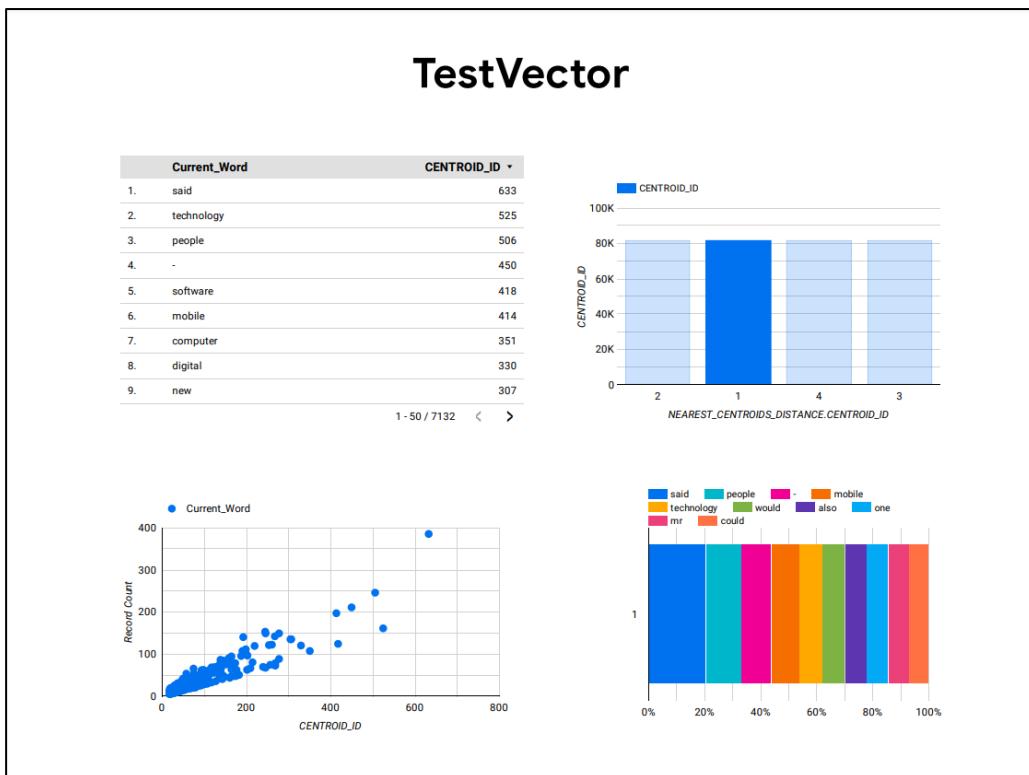


Figure 47: Information about clusters for testvector.csv generated by GCP ML

Information about cluster for trainvector.csv generated by GCP ML

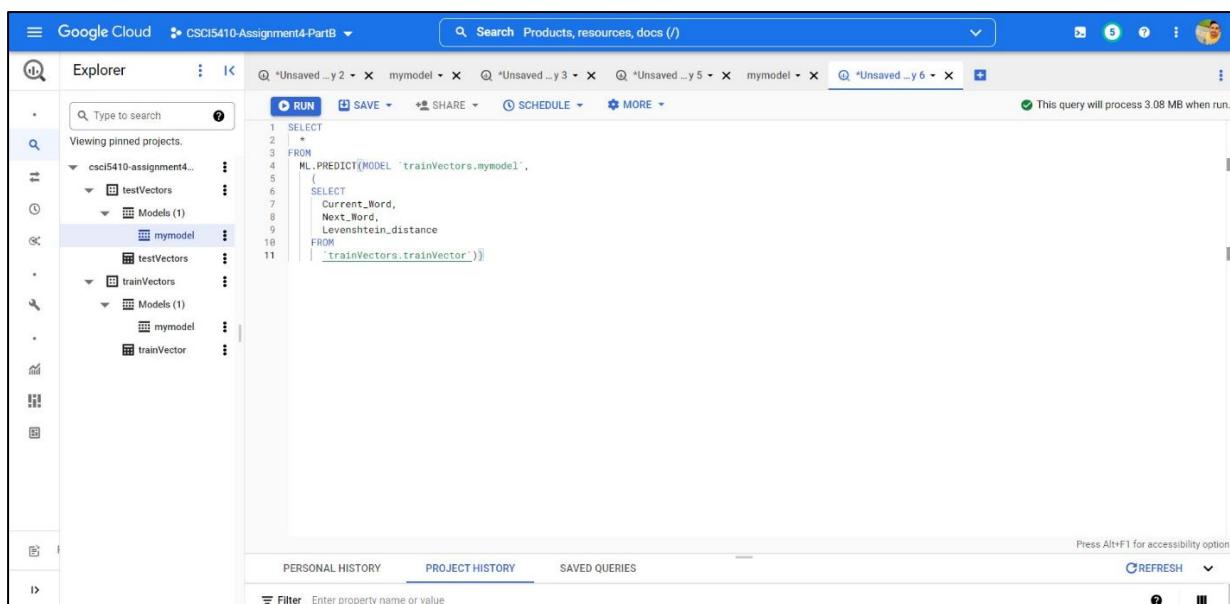


Figure 48: Information about clusters for trainvector.csv generated by GCP ML

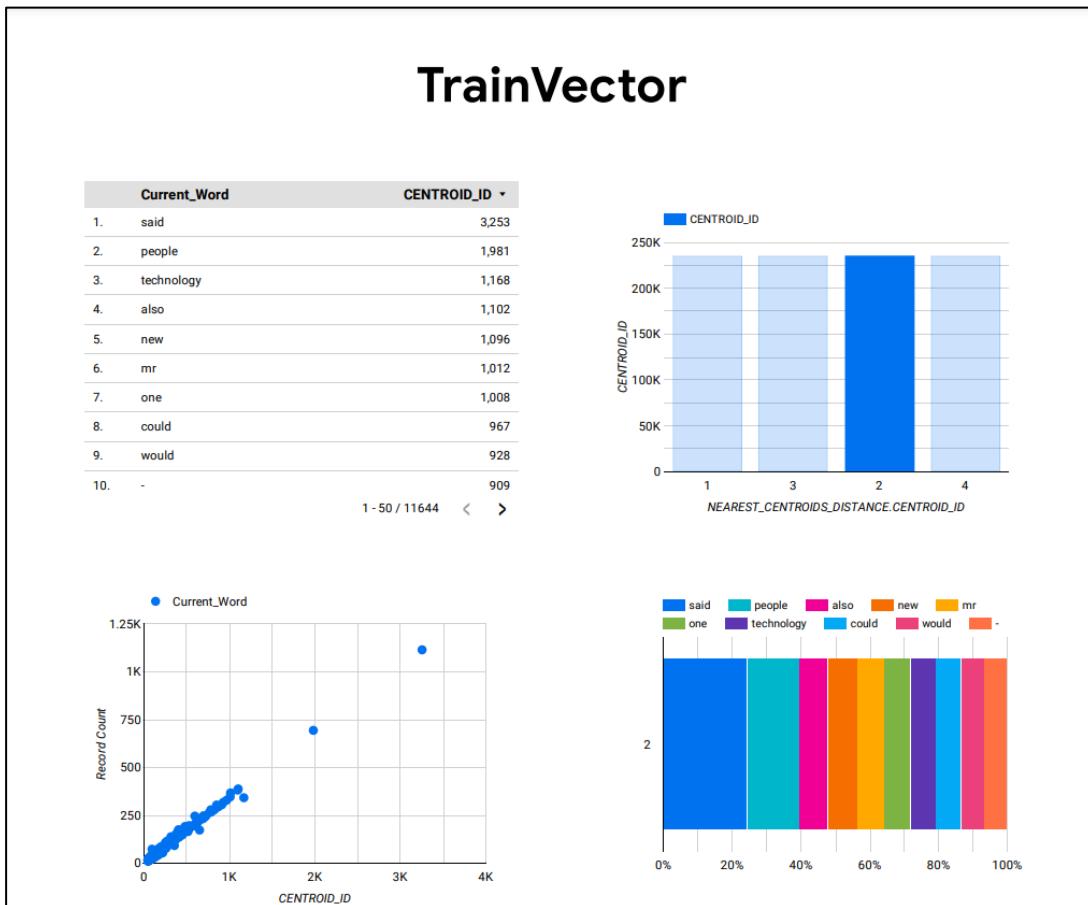


Figure 49: Information about clusters for trainvector.csv generated by GCP ML

Program/Scripts

GCSConnection.java

This java class is used to establish connection with the [Google Cloud Storage](#)

```
import com.google.auth.Credentials;
import com.google.auth.oauth2.GoogleCredentials;
import com.google.cloud.storage.Storage;
import com.google.cloud.storage.StorageOptions;
import java.io.FileInputStream;
import java.io.IOException;

public class GCSConnection {
    public Storage establishConnection() {
        try {
            Credentials credentials = GoogleCredentials.fromStream(new
FileInputStream("csci5410-assignment4-partb-bbfabf0ee9a5.json"));
            return
StorageOptions.newBuilder().setCredentials(credentials).setProjectId("csci5410-assignment4-
partb").build().getService();
        } catch (IOException e) {
            System.out.println("Error while creating GCP Connection: " + e.getMessage());
        }
        return null;
    }
}
```

DatasetProcessing.java

This class is used to create a bucket and then upload files in that bucket on [Google Cloud Storage](#)

```
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import com.google.cloud.storage.BlobId;
import com.google.cloud.storage.BlobInfo;
import com.google.cloud.storage.BucketInfo;
import com.google.cloud.storage.Storage;
public class DatasetProcessing {
    GCSConnection gcsConnection = new GCSConnection();
    private static final String SOURCE_BUCKET = "sourcedatab00899516";

    public boolean checkIfBucketExists() {
        Storage storage = gcsConnection.establishConnection();
        if(storage.get(SOURCE_BUCKET) == null){
            return false;
        } else return storage.get(SOURCE_BUCKET).exists();
    }

    public Storage createBucket() {
        Storage storage = gcsConnection.establishConnection();
        if (storage != null) {
            if (!checkIfBucketExists()) {
                storage.create(BucketInfo.of(SOURCE_BUCKET));
                System.out.println("Bucket " + SOURCE_BUCKET + " created
successfully.");
            }
        }
    }
}
```

```
        } else {
            System.out.println("Bucket " + SOURCE_BUCKET + " exists already.");
        }
    }
    return storage;
}
private void uploadFilesToGCSBucket(Storage storage) {
    try {
        File[] files = new File("Dataset/Test/").listFiles();
        if (files == null) {
            return;
        }
        System.out.println("\nUploading Files to "+SOURCE_BUCKET+"\n");
        for (File file : files) {
            System.out.println("Dataset/Test/" + file.getName());
            storage.create(BlobInfo.newBuilder(BlobId.of(SOURCE_BUCKET,
file.getName())).build(), Files.readAllBytes(Paths.get("Dataset/Test/" +
file.getName())));
            System.out.println(file.getName() + " has been uploaded to bucket " +
SOURCE_BUCKET + ".\n");
        }
    } catch (final IOException e) {
        System.out.println("Error while uploading file: " + e.getMessage());
    }
}
public void execute(){
    Storage storage = createBucket();
    uploadFilesToGCSBucket(storage);
}
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        DatasetProcessing datasetProcessing = new DatasetProcessing();
        datasetProcessing.execute();
    }
}
```

vectorGenerator.js

```

package vectorgenerator;
import com.google.api.gax.paging.Page;
import com.google.cloud.functions.BackgroundFunction;
import com.google.cloud.functions.Context;
import com.google.cloud.storage.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.stream.Collectors;
import java.util.stream.Stream;
public final class VectorGenerator implements
BackgroundFunction<VectorGenerator.GCSEvent> {
    private static final String TRAIN_VECTOR_CSV = "trainVectors.csv";
    private static final String TRAIN_DATA_BUCKET = "trainindatab00899516";

    /**
     * Reference: https://www.baeldung.com/java-levenshtein-distance
     * Reference: https://www.baeldung.com/java-csv
     */
    @Override
    public void accept(GCSEvent event, Context context) {
        Storage storage = StorageOptions.getDefaultInstance().getService();
        Page<Blob> blobs = storage.get(event.bucket()).list();
        ArrayList<String[]> dataLinesList = new ArrayList<>();
        StringBuilder listOfWords = new StringBuilder();

        for (Blob blob : blobs.iterateAll()) {
            String txtFileContent = regex(blob);

            ArrayList<String> wordsList = new
ArrayList<>(Arrays.asList(txtFileContent.toLowerCase().split(" ")));
            wordsList.removeAll(Arrays.asList("she", "her", "hers", "herself", "it", "its",
"itself", "they", "them", "their", "theirs", "themselves", "what", "which", "before",
"after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over",
"under", "again", "further", "then", "once", "here", "there", "when", "where", "why",
"how", "all", "any", "both", "each", "few", "more", "who", "whom", "this", "that",
"these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have",
"has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but",
"if", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about",
"against", "between", "into", "through", "during", "most", "other", "some", "such",
"no", "nor", "not", "only", "own", "same", "so", "than", "too", "very", "s", "t", "can",
"will", "just", "don", "should", "now"));
            for (String word : wordsList) {
                listOfWords.append(word).append(" ");
            }
            /**
             * Reference: https://www.baeldung.com/java-csv
             */
            for (int i = 0; i < wordsList.size() - 1; i++) {
                String currentWord = wordsList.get(i);
                String nextWord = wordsList.get(i + 1);
                int[][] dp = new int[currentWord.length() + 1][nextWord.length() + 1];
                for (int i1 = 0; i1 <= currentWord.length(); i1++) {
                    for (int j = 0; j <= nextWord.length(); j++) {
                        if (i1 == 0) {
                            dp[i1][j] = j;
                        } else if (j == 0) {
                            dp[i1][j] = i1;
                        } else {
                            /**
                             * Reference: https://www.baeldung.com/java-levenshtein-distance
                             */
                            dp[i1][j] = Arrays.stream(new int[]{dp[i1 - 1][j - 1] +
                                (currentWord.charAt(i1 - 1) == nextWord.charAt(j - 1) ? 0 : 1),
                                dp[i1 - 1][j] + 1, dp[i1][j - 1] + 1}).min().orElse(Integer.MAX_VALUE);
                        }
                    }
                }
            }
        }
    }
}

```

```
        int distance = dp[currentWord.length()][nextWord.length()];
        dataLinesList.add(new String[]{currentWord, nextWord,
String.valueOf(distance)}));
    }
}
StringBuilder sb = new StringBuilder();
for (String[] strings : dataLinesList) {
    sb.append(Stream.of(strings)
        .map(data -> {
        /**
         * Reference: https://www.baeldung.com/java-csv
         */
        String escapedData = data.replaceAll("\n", " ");
        if (data.contains(",") || data.contains("\")" || data.contains("\"")) {
            data = data.replace("\"", "\\"\"");
            escapedData = "\"" + data + "\"";
        }
        return escapedData;
    })
        .collect(Collectors.joining(","))
    ).append("\n");
}
if (isbucketNull(storage)) {
    storage.create(BucketInfo.of(TRAIN_DATA_BUCKET));
} else if (isBucketExists(storage)) {
    storage.create(BucketInfo.of(TRAIN_DATA_BUCKET));
} else {
}
storage.create(BlobInfo.newBuilder(BlobId.of(TRAIN_DATA_BUCKET,
TRAIN_VECTOR_CSV)).build(), sb.toString().getBytes());
}

/**
 * Reference: https://www.baeldung.com/java-csv
 */
private String regex(Blob blob) {
    return new String(blob.getContent())
        .replaceAll("[,\.\-\!\\"{}?!@#$%^&*\[\]\]+", "")
        .replaceAll("[\s]+", " ");
}

private boolean isBucketExists(Storage storage) {
    return !storage.get(TRAIN_DATA_BUCKET).exists();
}

private boolean isbucketNull(Storage storage) {
    return storage.get(TRAIN_DATA_BUCKET) == null;
}

public static class GCSEvent {
    String bucket;
    String name;
    String metaGeneration;

    public GCSEvent(final String bucket,
                   final String name,
                   final String metaGeneration) {
        this.bucket = bucket;
        this.name = name;
        this.metaGeneration = metaGeneration;
    }
}
}
```

References

- [1] Google, "Cloud Functions," Google, [Online]. Available: <https://cloud.google.com/functions>. [Accessed 09 July 2022].
- [2] Baeldung, "How to calculate Levenshtein Distance in Java?" Baeldung, [Online]. Available: <https://www.baeldung.com/java-levenshtein-distance> [Accessed 09 July 2022].
- [3] Wikipedia, "Levenshtein Distance," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Levenshtein_distance [Accessed 09 July 2022].
- [4] Google, "Cloud Storage," Google, [Online]. Available: <https://cloud.google.com/storage>. [Accessed 09 July 2022].
- [5] Google, "Vertex AI," Google, [Online]. Available: https://cloud.google.com/vertex_ai/docs/reference/rest [Accessed 09 July 2022].
- [6] Google, "Google Cloud," Google, [Online]. Available: <https://cloud.google.com/gcp>. [Accessed 09 July 2022].
- [7] Google, "Operation Suits," Google, [Online]. Available: <https://cloud.google.com/logging>. [Accessed 09 July 2022].
- [8] javaTpoint, "K-Means Clustering Algorithm," javaTpoint, [Online]. Available: <https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning> [Accessed 09 July 2022].
- [9] Wikipedia, "Elbow method (clustering)," Wikipedia, [Online]. Available: [https://en.wikipedia.org/wiki/Elbow_method_\(clustering\)](https://en.wikipedia.org/wiki/Elbow_method_(clustering)) [Accessed 09 July 2022].
- [10] Baeldung, "How to Write to a CSV File in Java," Baeldung, [Online]. Available: <https://www.baeldung.com/java-csv> [Accessed 09 July 2022].
- [11] Wikipedia, "k-means clustering," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Levenshtein_distance [Accessed 09 July 2022].
- [12] GitHub, "NLTK's list of english stopwords," GitHub, [Online]. Available: <https://gist.github.com/sebleier/554280> [Accessed 09 July 2022]