

# Cloud Computing: Assignment Two



David Sarkies [S3664099]

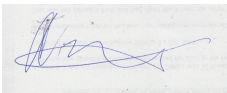

Max Ardas [S3717924]

<https://beercoffeemaps.com>

[https://github.com/s3664099/CloudComputingAssignment02\](https://github.com/s3664099/CloudComputingAssignment02)

(Please note that the repository is private)

## Contribution Agreement

Name: David Sarkies	Name: Max Ardas
Student ID: S3664099	Student ID: S3717924
Contributions: <ol style="list-style-type: none"><li>1. Configuration of Cloud SQL</li><li>2. Python, HTML/Jinja, Javascript</li><li>3. Configuration of Cloud Storage</li><li>4. Configuration of Custom Domain and SSL certificates</li><li>5. CSS</li><li>6. Configuration of Google App Engine</li><li>7. Configuration of Google Translate</li><li>8. Configuration of traffic splitting</li></ol>	Contributions: <ol style="list-style-type: none"><li>1. Python, HTML/Jinja, Javascript application programming.</li><li>2. Configuration of Datastore.</li><li>3. User authentication.</li><li>4. Logging configuration and routing into BigQuery.</li><li>5. Email configuration.</li></ol>
Percentage: 50%	Percentage: 50%
<p><i>By signing below, I certify all information is true and correct to the best of my knowledge.</i></p> <p>Signature: </p> <p>Date: 24/05/2020</p>	<p><i>By signing below, I certify all information is true and correct to the best of my knowledge.</i></p> <p>Signature: </p> <p>Date: 17/05/2020</p>

# Introduction

## Summary:

The objective of this project is to create a simplified and graphical review platform for users to share their experience of businesses, particularly when it comes to coffee and beer, as well as maintain a log of places they have visited.

## Project Motivations:

The motivation behind this project was to create a location review platform that isn't bounded by the countries that the app is active in, or isn't bound to a particular type of establishment (such as restaurants or pubs). Other similar platforms also lack the functionality for a user to quickly sort through their location history to identify places they have already been and what they thought of those places, which may affect decisions about which places a user will visit in the future. Another problem with many of the currently available apps is that the main focus is on the reviews as opposed to being on the maps, and as such it is difficult to view such places on a map.

With online reviews becoming a prevalent source of information for customers, it is important that reviews are simple to digest and are authentic. This motivated the simple formatting and representation of reviews in the application, as well as having the map as the first point of call.

## Functionality:

The project allows users to add their current location to a places database so that they and other users are able to provide ratings for a location. The project currently supports four types of locations: pubs, takeaway shops, cafes and museums. Once added, users are able to write and view reviews for a given location. Reviews are bound to the coordinates of the place, as well as the user's unique account identification key, meaning an account must be created by the user before they are able to write reviews.

Locations added to the places database are shown on the main page of the application on a Google Map. Users who are logged in are then able to view places that they have already visited and reviewed. The user is also able to narrow down the places, at this stage, to bars, cafes, or both bars and cafes. In order to support an international community of reviews, the project also supports translation to three languages other than English: French, Italian and German.

Multiple versions have also been deployed, which focus on certain areas. One has been deployed for New York, one for San Francisco, one for Hong Kong, for Rome, for London, for Frankfurt, for Paris, and the final one for London. The versions deployed for Frankfurt, Paris, and Rome have been configured for the appropriate language. The traffic is split based on the user's IP address, however at this stage traffic isn't split based upon location.

### Real Life Use Case:

The project can be used as a real life application by people who like to provide reviews for businesses and travel frequently. It will allow users to maintain a “journal” of places they have visited over time and what they thought of their experience at a particular place. For businesses, it is also a way to facilitate authentic reviews from real customers and for customers to share their experience with other users of the application, which can allow them to make informed decisions about places to visit and whether the places are suitable for them.

### Advantages of the Application:

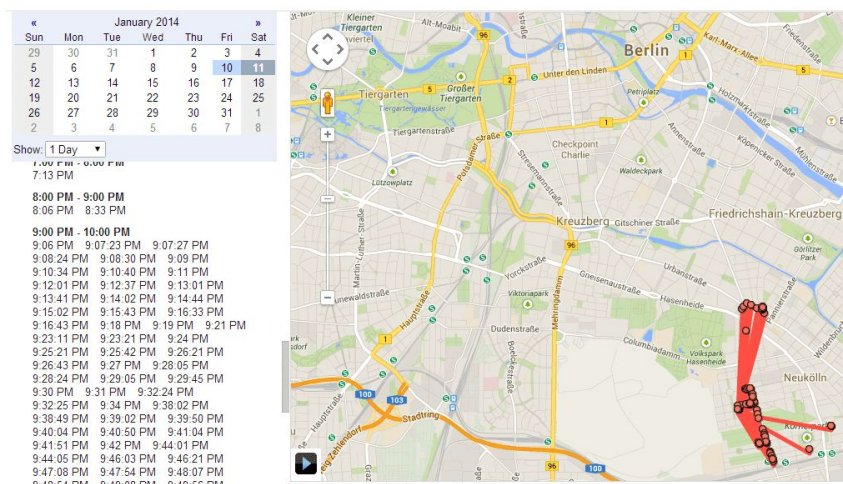
The main advantages of this application over related works is the simple graphical representation of places users have visited through the use of filtering and Google Maps. The structure of the reviews is also very simple as it limits reviews to a like or dislike and a short textual description. This is superior to the current rating systems since ratings can be rather difficult to determine. Instead, by simply having a like/dislike option makes the decision a lot easier. Adding places to the database is also the responsibility of the users, so businesses do not have to go through a verification process.

## Related Works

### Google Maps:

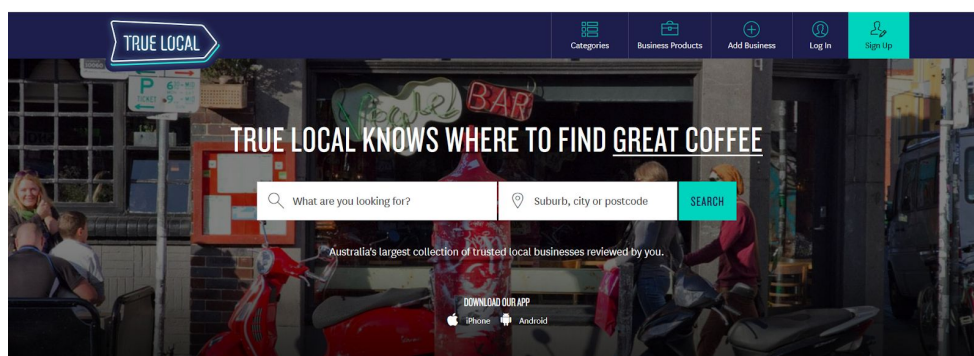
Google Maps is known (perhaps infamously) for making heavy use of location services on devices, and use this to construct a comprehensive location history of its users. While this feature is somewhat useful for small amounts of places, it does not scale well to larger amounts as they become slow and cumbersome to sort through. The platform also allows users to leave reviews for places. The other problem with Google Maps is that it can be difficult to visualise a list of places the user has reviewed. While a small number of places works, it becomes difficult where users have reviewed a significant amount of places, and to get a visualisation of them all not only takes a tremendous amount of time, it also slows the computer down significantly.

Location history



### True Local/Yelp:

True Local and Yelp are two well known sites for listing business information and reviews, however their shortfall is that they are limited to only a select group of countries. True Local also has a search function that allows you to narrow your business search to a particular type and location. Also, unlike our application, the main focus is on the reviews as opposed to the maps. As such when the user visits the site, the first page that is visible is the search page as opposed to a map page.



## Tripadvisor/Zomato

Similarly to True Local and Yelp, Tripadvisor and Zomato are well established platforms for users to review businesses and businesses to list their details. However, these platforms are limited in the type of businesses that they specialise in. For example, Zomato is focused on food and drink (serving only twenty-four countries) and Tripadvisor is focused on travel experiences, accommodation and transportation. The other problem with Tripadvisor is while it is a popular site, the actual reviewing part of the site is quite difficult to navigate, and inconsistent across the three types of places to review. It is the author's opinion (David Sarkies) that the User Interface from Tripadvisor really isn't the best.

# Software Design/Architecture

## Runtime and Framework:

Beer and Coffee Maps operates as a Python 2.7 web app running on the WebApp2 Framework and Google App Engine. The features of WebApp2 and GAE are accessed using their Python client libraries, which can be installed through Pip, the Python package manager. WebApp2 was designed specifically for use with Google App Engine and is responsible for handling requests, session storage and URL routing.

Traffic splitting by IP Address has also been enabled to route users to different versions of the app. This is to assist with load balancing, but it also sends the user to a version that has been configured slightly differently, based on the first location that appears, and also the default language that has been set up.

Google App Engine also intrinsically supports other cloud services, such as emailing and the ability to send logs to Google Cloud Platform, as well as automatic scaling.

## APIs

The application relies on some Application Programming Interfaces in order to provide functionality. The main API utilised is the Google Maps Javascript API, which allows the programmer to generate a Google Map and place it onto a webpage, as well as add points to the map. The map is generated by sending requests to the API via Javascript.

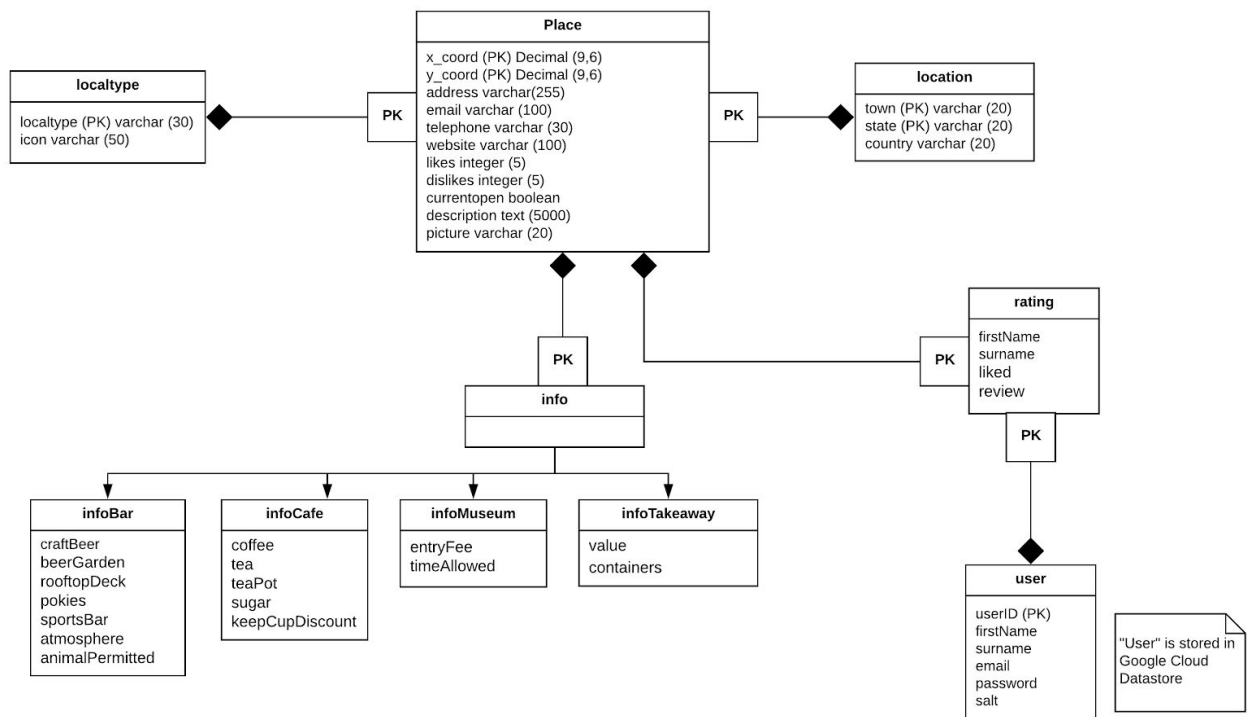
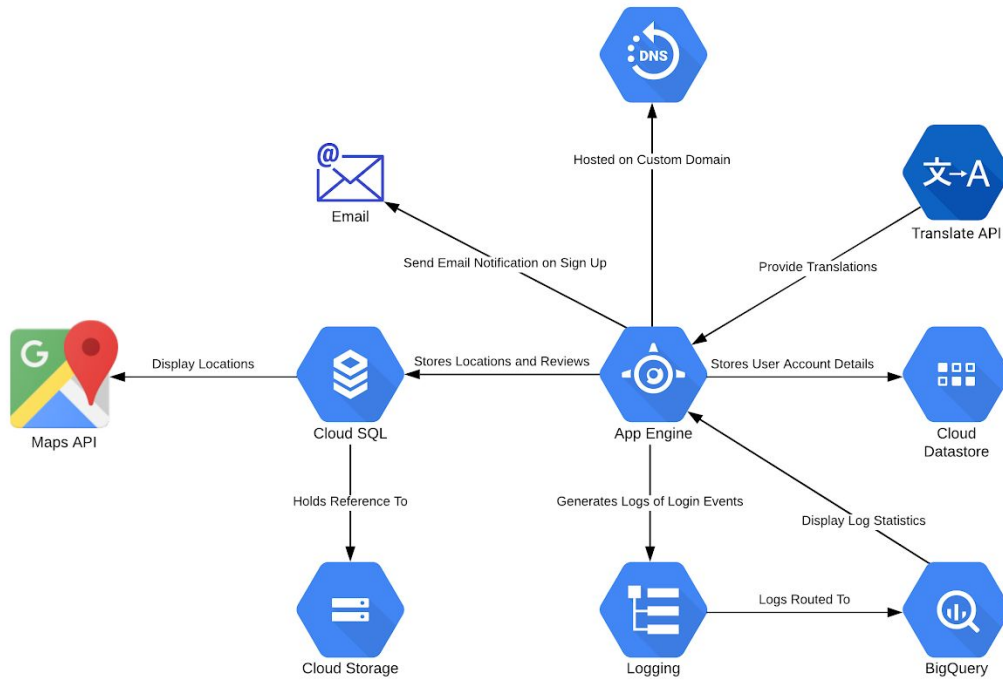
Other APIs were accessed through the use of client libraries and could be installed via Pip similarly to WebApp2. These include Google Translate and BigQuery. BigQuery has the capability of handling massive amounts of data, meaning it contributes to the scalability of the application.

## Storage:

The application consists of three storage components. The first is Cloud Datastore, which is a No-SQL based data storage and is used to store user details. The second is a Google Cloud MySQL instance, and is used to store place information and reviews. A schema of the database is included below. The information in this database is accessed through queries constructed using the Python MySQL library. The final storage component are two cloud storage buckets that are used to store map icons and place photographs. The MySQL database holds references to icons and the photos in these buckets. A third bucket is also used to hold images, such as the title image, for the website.

## Custom Domain:

The application is also hosted on a custom domain. The HTTPS protocol is enabled through a Google provisioned SSL certificate for App Engine applications.





# Implementation

## Getting Started

Firstly, create a new Google Cloud Platform project at [this page](#) and clone the Beer Coffee Maps project from the [GitHub repository](#) onto your machine. After doing that, refer to the “Setting Up Authentication” of [this](#) article to create a service account and obtain authentication credentials in order to access the project and use client libraries. The article will also guide you through setting up the Google Application Credentials environment variable.

Ensure that the Google Cloud SDK is installed on your system by following [this link](#).

## Enabling APIs/Getting API Keys

For the project to work correctly, you will need to enable some APIs in the Google Cloud Console, as well as obtain an API key for Google Maps. Navigate to [this page](#) to get started. In the search box, search for “Maps Javascript API” and select “enable”. After enabling this API, navigate to the [credentials](#) page of the Cloud Console. Select “create credentials” and select “API key”. Copy this API key and paste it into the following section of index.html:

```
<script>{% include 'template.js' %}</script>
<script src="js/script.js"></script>

<script src="https://maps.googleapis.com/maps/api/js?key=**YOUR MAPS API KEY**&callback=initMap"
async defer></script>
```

Navigate back to the [API Library](#) page and then search for the BigQuery API, then enable it. Repeat the process for the Translation API. These APIs do not require that API keys be generated as it will use the credentials stored in the environment variable configured above.

## Setting Up Translations

- 1) **Please be aware that the Google Translate API does not exist for python 2. As such any functions required access to the Translate API needs to be run using python 3.** This section, and the next section, are set up using python 3.
- 2) To set up [Google Translate](#) you need to activate it in the project. The API is located in the Artificial Intelligence collection of services. You will need to activate it here. Once the translation API has been activated, you need to create a service account to obtain an API key. Once the service account has been created, you will need to download the private key as a JSON file. Once it has been downloaded, you will need to set the GOOGLE\_APPLICATION\_CREDENTIALS to the path where the key is located. For Mac and Linux users the command is as follows:

```
export GOOGLE_APPLICATION_CREDENTIALS="[PATH]"
```

For windows users the command is as follows:

```
$env:GOOGLE_APPLICATION_CREDENTIALS="[PATH]"
```

[PATH] is the location of the private key.

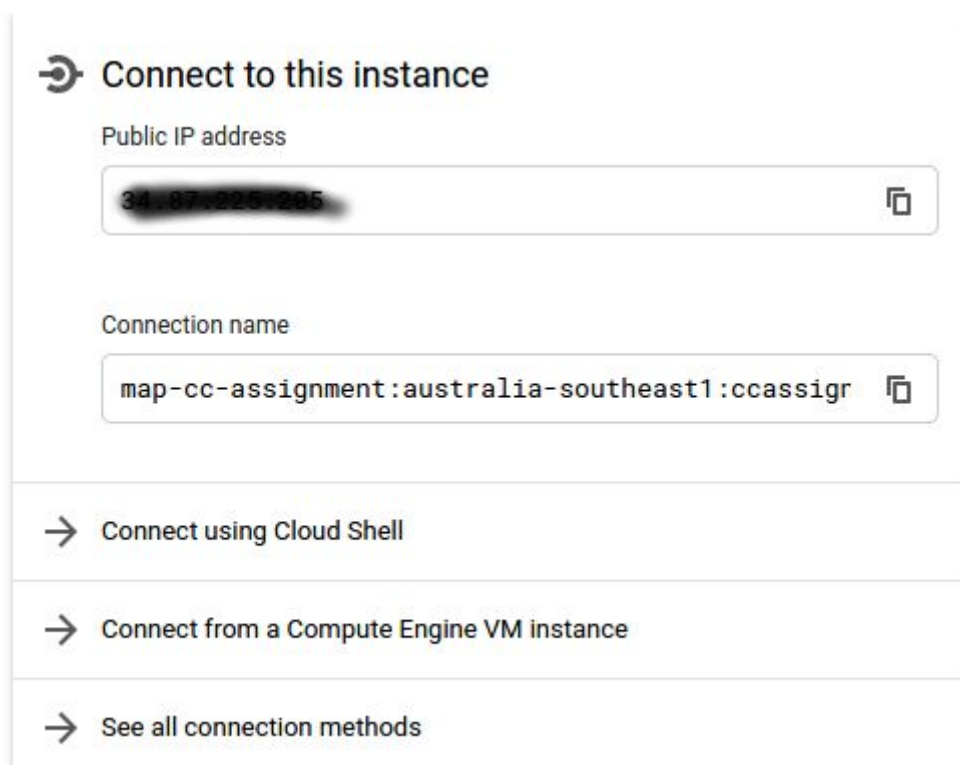
- 3) Once this has been done, you now need to install the python translate library, which is done using the following command:


```
pip install google-cloud-translate==2.0.1
```

- 4) Once this is done you are now able to perform translation functions on your machine.


### Setting Up Cloud SQL

- 1) The script for setting up the database is located in the folder Database Setup, and contains the file databaseSetup.py. Also in this folder, in a .csv format, is the data containing details of the places that have been added to the application. However, it should be noted that this file will need to be run using python3 since the script calls upon the Google Translation API to provide translation services for the place descriptions. The script only loads up place name data, and does not add any reviews, however it does translate the descriptions.
- 2) However, before this can be done an SQL instance needs to be set up. To do that you select the SQL tab and it takes you to the main SQL page which has a 'Create Instance' button at the top. Select that and then select MySQL. Here you will be asked to enter an instance Id and a Root password. Fill them in, and remember them since you will need to use them later. Leave the others as default (though changing the location might help with speed). Now that the instance has been created you need to create the database. Go to the Databases tab and selection 'Create Database'. Type in the name of the database, and keep a copy of the name for further use.
- 3) Next, go to Overview and look for the following box:





 **Connect to this instance**


Public IP address




Connection name



 **Connect using Cloud Shell**

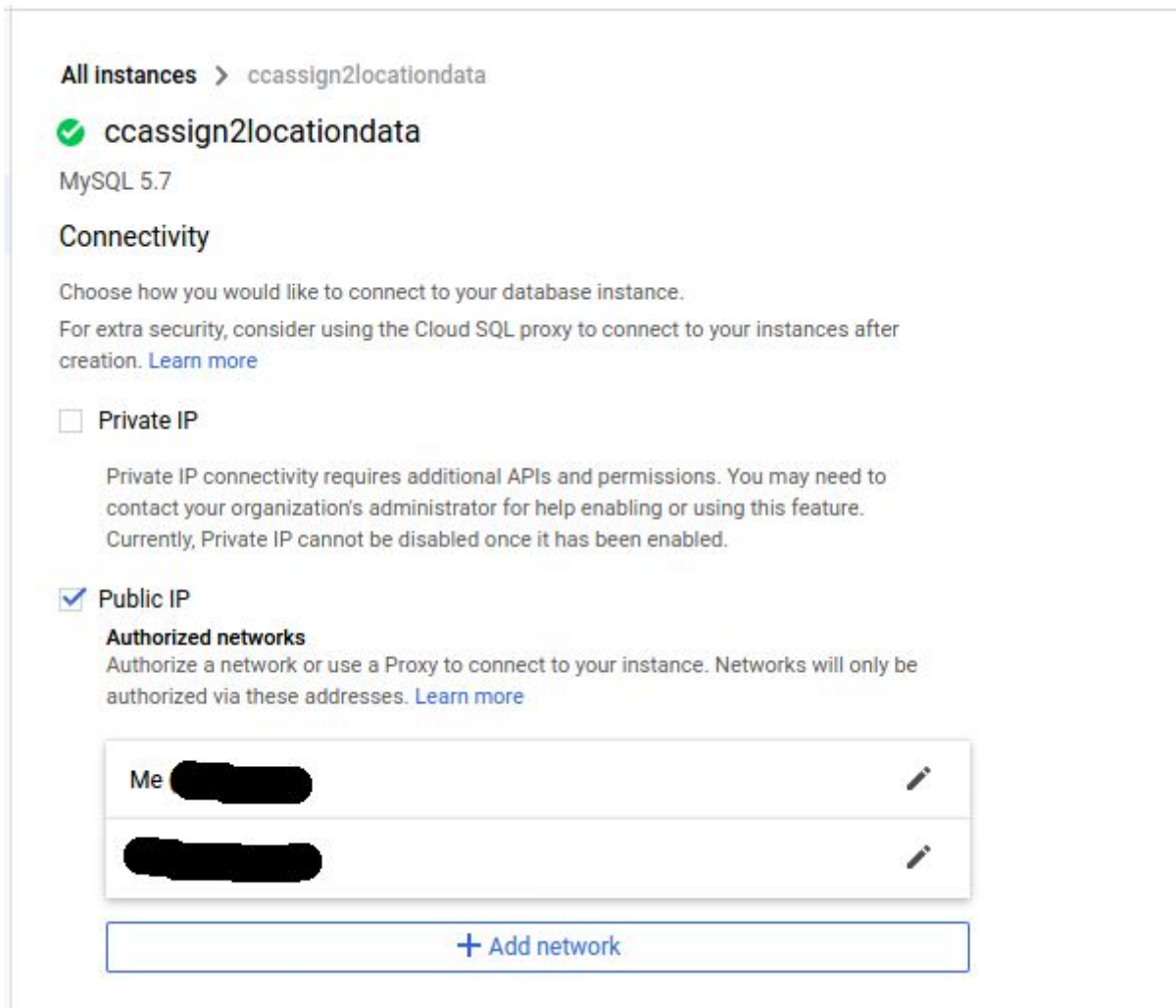
 **Connect from a Compute Engine VM instance**

 **See all connection methods**

- 4) Copy the public IP address as you will need that to connect to the instance.
- 5) Next, you will need to get your IP address, and you can do this with the following command:

```
curl icanhazip.com
```

- 6) Now that you have your IP address, click on the tap that says 'Connections' and you should see something like this:



The screenshot shows the AWS Management Console interface for a MySQL instance. At the top, it says 'All instances > ccassign2locationdata'. Below this, there's a green checkmark icon followed by the instance name 'ccassign2locationdata' and 'MySQL 5.7'. The 'Connectivity' section is active, with the heading 'Choose how you would like to connect to your database instance.' and a note about using the Cloud SQL proxy for extra security. There are two radio button options: 'Private IP' (unchecked) and 'Public IP' (checked). Under 'Public IP', there's a section for 'Authorized networks' with a description and a 'Learn more' link. Below this, there's a list of authorized networks, each with a name (one is 'Me'), a redacted IP address, and an edit icon. At the bottom, there's a button labeled '+ Add network'.

- 7) Select 'Add Network' and the following box will open:

- 8) This will enable you to connect to the database externally so that you are able to upload the data. Enter a name (which is optional), and then place the IP address in the box that says network. Click Done and Save, and your IP address will now be authorised.
- 9) Next, go to Users and select 'Create User Account', and enter your name and password. Remember these for the next, and final, step as this also enables you to connect from the database remotely.
- 10) Finally, open up the databaseSetup.py file in your preferred IDE/Editor and at the top of the page you should see the following:

```
#This is required for python3 to create and manipulate mySql databases
import pymysql
from google.cloud import translate_v2 as translate
translate_client = translate.Client()

hostname = '
username = '
password = '
database = '
datafile = 'places.txt'
text = "Hello"
```

Where I have blacked out the names, this is where you will need to enter the details.

The **hostname** is IP address you collected from Connect to this Instance.

The **username** is the username you selected when you created the user account.

The **password** is the password you created for the same account.

- 11) Finally, the database field is where you put the name of the database. *Not* the name of the instance, but the name of the database you chose when you created the database.

- 12) Now that this is done, you can run the file by typing `python3 databaseSetup.py`, and go away and do something else while the database is created and uploaded. This will take some time, particularly with the translation functions.

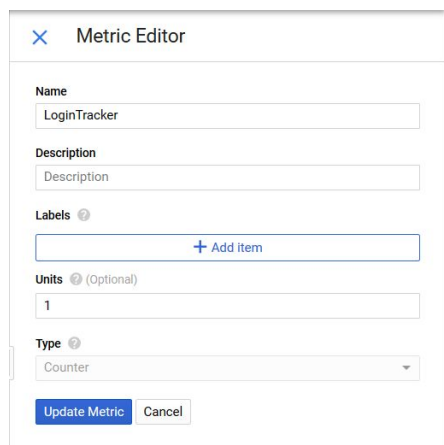
## Setting Up Logging

```
if (pwdhash == storedPass):  
    text = hashedEmail + " logged into the application."  
    logging.info(text)
```

- 1) In this project, login event logs are sent to the standard info level logger. The output of these logs can be viewed in the “Logs Viewer” section of the Google Cloud Platform when you have the project selected.
- 2) In order to send these logs to a BigQuery dataset, a Logs-based Metric needs to be created. To do this, navigate to the Logs-based Metrics section of the Logging dashboard and select “create metric”. In the dropdown menu of the search/filter box, select “convert to advanced filter” and copy the following filter:

```
1 resource.type="gae_app"  
2 logName=("projects/**YOUR PROJECT ID**/logs/appengine.googleapis.com%2Frequest_log")  
3 protoPayload.line.logMessage: ("logged into the application")
```

- 3) Now, fill in the metric editor like so:



- 4) After creating the metric, on the same page, select “create sink”. Fill in the sink editor like so:

- 5) When prompted for a sink destination, select “create new BigQuery dataset” and provide a name. Now logs will be routed into a BigQuery dataset, which can now be queried by the application.
- 6) In the main.py file, modify the query so it contains your GCP Project ID and BigQuery dataset name:

```
query = """
SELECT count(*) FROM `**PROJECT ID**.**BIGQUERY DATASET NAME**.appengine_googleapis_com_request_log_****` + tablecode + """
where DATE(timestamp) = CURRENT_DATE LIMIT 1000;
"""
query_job = client.query(query)
results = query_job.result()
```

- 7) Now the amount of daily logins will be displayed on the main page of the application.

### Setting up a Custom Domain Name

- 1) To set up a custom domain, you first of all need to get a domain name. While there are numerous places where you can register domain names, [Google Domains](#) is reasonably user friendly, and also enables you to easily port the domain name into the application. Find a domain name, and pay the yearly fee to register it.
- 2) Once you have done that, go to App Engine and then to settings. Now, the domain name needs to have been registered using the same email address that you are using for the project. If you aren't all is not lost as you can add another email address to the domain name in Google Domains. Anyway, you should see the following, and if you don't, click on 'Custom Domains'.

Google Cloud Platform Cloud Computing Assignment 2 Search products and resources

App Engine Settings

Application settings Custom domains SSL certificates Email senders

Add a custom domain Enable managed security Disable managed security

All domains mapped to this application are shown below. Only owners of a domain may remove one of its mappings.

Custom domain name	SSL security	Certificate ID	Record type	Data	Alias
beercoffeemaps.com	Google-managed, auto-renewing	-	A	216.239.32.21	(none)
			A	216.239.34.21	
			A	216.239.36.21	
			A	216.239.38.21	
			AAAA	2001:4860:4802:32::15	
			AAAA	2001:4860:4802:34::15	
			AAAA	2001:4860:4802:36::15	
			AAAA	2001:4860:4802:38::15	
www.beercoffeemaps.com	Google-managed, auto-renewing	-	CNAME	ghs.googlehosted.com.	www

- 3) Now, select Add a Custom Domain, and then follow the steps. If you have registered the domain through Google Domains, and used the same email address, then verification will be quite quick. However, if you have registered the Domain Name elsewhere then the verification is a little trickier. Once you have done that, you will need to update the DNS settings that have been provided by App Engine. You will need to select DNS in Google Domains, go to the bottom, and enter the details there.

### Custom resource records

Resource records define how your domain behaves. Common uses include pointing your domain at your web server or configuring email delivery for your domain. [Learn more](#)

@ A 1H IPv4 address + Add

Name	Type	TTL	Data		
@	A	1h	216.239.32.21	Delete	Edit
@	AAAA	1h	2001:4860:4802:32::15	Delete	Edit
www	CNAME	1h	ghs.googlehosted.com.	Delete	Edit

- 4) Finally, return to the App Engine settings page, and select 'Enable Managed Security' and select the domain names you wish this to apply to. What this does is that Google will then manage the SSL certificates, automatically giving the website https status.

### Setting Up Emailing

- 1) Towards the end of the SignUpPage class in main.py, there is a method call for sendNewAccMail. Replace the first parameter (the sender address) with the following:



```
sendNewAccMail("noreply@**YOUR PROJECT ID**.appspotmail.com", email, firstName, surname)
```

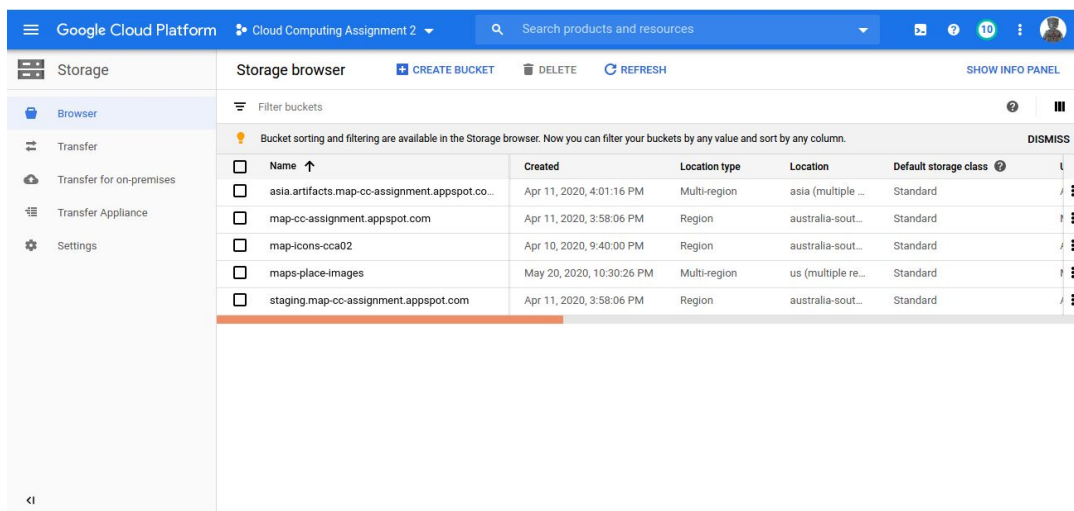
- 2) Replace the email message in the body of sendNewAccMail with the following:

```
def sendNewAccMail(senderAdd, recieverAdd, firstName, surname):  
    mail.send_mail(sender = senderAdd,  
        to = firstName + " " + surname + " <" + recieverAdd + ">",  
        subject = "Welcome!",  
        body = "Welcome " + firstName + ""!  
  
        Thanks for signing up to the app. Visit **LINK TO YOUR HOSTED PROJECT** to sign in!""")
```

- 3) Now the project will send an email from the specified address to the user after they sign up to the application. The message may be sent to your spam folder, depending on your email provider. The subject and body of this email can be customised in any way.
- 4) It is also possible to add a custom email address as well, such as **noreply@beercoffeemaps.com**. If you set up your domain through Google Domains you can then link the domain name to an email address. However, to do this you will need to set up a G-Suite account. It was decided not to do this due to the costs involved. However, G-suite does offer the option of a free trial, though this was not taken up.
- 5) Anyway, once you have set up your email in G-Suite, navigate to settings in App Engine and select email senders. Select add, and enter the authorised email address. This email address can now be used to send emails from the application.

## Setting Up Cloud Storage For Icons

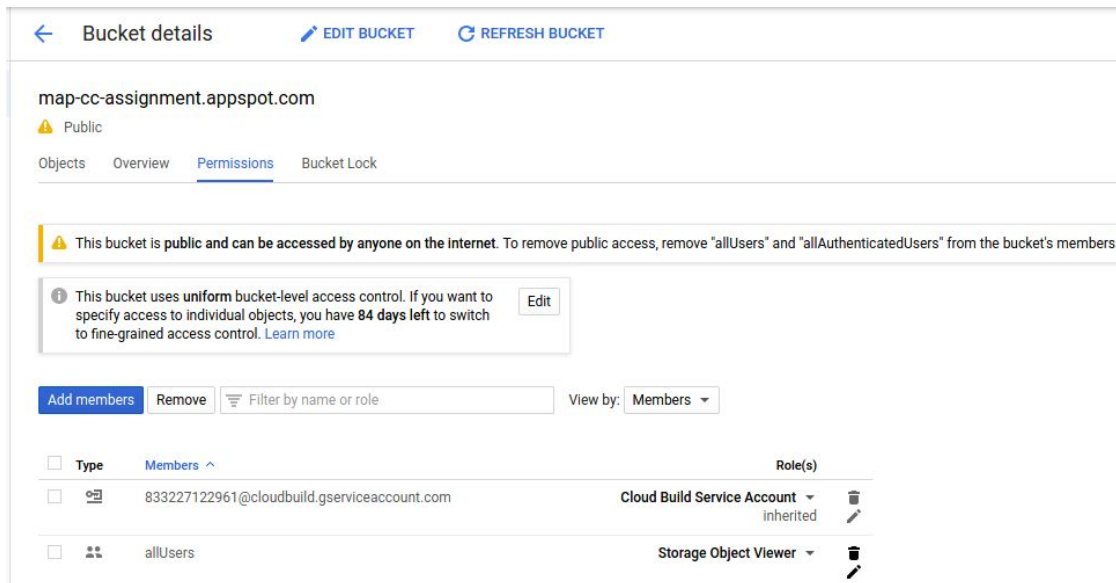
- 1) To set up the storage function for the icons navigate to storage in the side bar, and select browser. You will be taken to the following window:



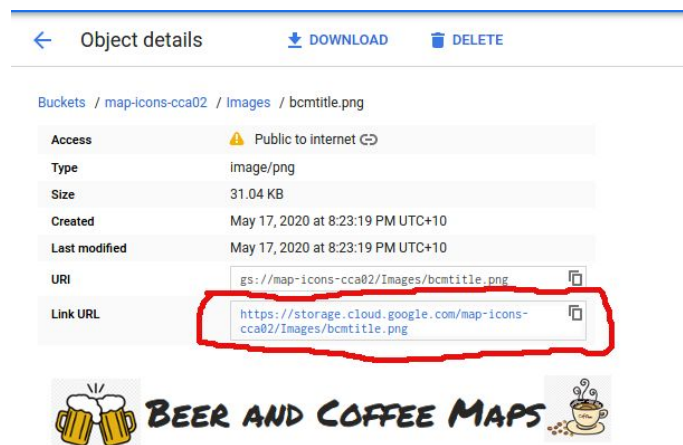
Name	Created	Location type	Location	Default storage class
asia.artifacts.map-cc-assignment.appspot.com	Apr 11, 2020, 4:01:16 PM	Multi-region	asia (multiple ...	Standard
map-cc-assignment.appspot.com	Apr 11, 2020, 3:58:06 PM	Region	australia-sout...	Standard
map-icons-cca02	Apr 10, 2020, 9:40:00 PM	Region	australia-sout...	Standard
maps-place-images	May 20, 2020, 10:30:26 PM	Multi-region	us (multiple re...	Standard
staging.map-cc-assignment.appspot.com	Apr 11, 2020, 3:58:06 PM	Region	australia-sout...	Standard



- 2) Select 'Create Bucket' and follow the prompts. For the bucket name you will need to enter a unique identifier, and select the default option. The name is a global name, meaning that nobody else will be able to use this name.
- 3) Next, select the bucket, and then select permissions. You will see a screen like this:



- 4) Select 'Add Members' and enter allUsers. Google will give you a warning, but this is required to allow the images to be accessed by everybody using the application.
- 5) Next, navigate back to objects, and select create folder, which is used to hold the images. Navigate into that folder and select the images that you wish to upload to the folder. Once this has been done, you can then access the images. Another bucket has been created to hold the pictures of the locations, as well as the title image.
- 6) You will then need to click on the image to be able to find the link to that image. The following screenshot shows you what the image url is:



- 7) However, there is still a slight catch, namely the URL is incorrect and will only allow access to the images if you are signed into the account that is using the application. As such, you will need to change the URL as follows:

<https://storage.googleapis.com/map-icons-cca02/Images/bcmtitle.png?authuser=2>

- 8) Further, since you are using a different bucket to the one that the code has been configured for, you will need to change the addresses in the code that point to the bucket that is being used by our project. However, our buckets are open to the public, so you do not need to change any code, nor do you need to upload any images since you can simply use the ones that we have already made available.

## Deployment

- 1) Once everything is set up, the application can now be deployed to App Engine via the command line. Navigate to the root directory of the project that was cloned from the GitHub repository and open a command line window. Deploy the application as per [this guide](#).
- 2) The final thing that you need to do is to make the app a distributed model. First of all you may want to change some aspects of what you are deploying. To change the default language you will need to open the main.py file and go to the following line:

```
38 #This function sets up the jinja enviroment
39 JINJA_ENVIRONMENT = jinja2.Environment(
40     loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
41     extensions=['jinja2.ext.autoescape'],
42     autoescape=True)
43
44 #Establish a class to hold variables pulled from the static HTML page
45 class search():
46     locale_type = "All"
47     locale_place = "Sunbury"
48
49 class language():
50     language = 'en'
51
52 class show_locations():
53     selection = "everything"
54
55 class place():
56     lng = 0
57     lat = 0
58
```

The supported languages as: English (en), German (de), French (fr), and Italian (it).

- 3) To change the default location where the app opens up, you will need to navigate to the js folder, and open the file named script.js. The default co-ordinates for the map are the following lines:

```

1 //set up the variables
2 var markers = new Array();
3 var infowindows = new Array();
4 var iconSize = false;
5 var iconShown = false;
6 var smallIcon = 20;
7 var largeIcon = 40;
8 var longitude = -37.8165686;
9 var latitude = 144.9805071;
10
11 //initialise the Google Maps API
12 function initMap() {
13     locationStatusText = document.getElementById("locationStatus");
14
15     //gets the geolocation of the user
16     if (navigator.geolocation) {
17         navigator.geolocation.getCurrentPosition(function (position) {
18             var pos = {
19                 lat: position.coords.latitude,
20                 lng: position.coords.longitude
21             };
22

```

- 4) You can change these values and then deploy other versions of the app. Next navigate to App Engine and select the button labelled versions. You will see the following screen:

Version	Status	Traffic Allocation	Instances	Runtime	Environment	Size	Deployed	Diagnose
20200524t145809	Serving	12%	0	python27	Standard	26.2 MB	May 24, 2020, 3:01:43 PM by david.sarkies@internode.on.net	Tools
20200524t144513	Serving	12%	1	python27	Standard	26.2 MB	May 24, 2020, 2:45:58 PM by david.sarkies@internode.on.net	Tools
20200524t143405	Serving	12%	1	python27	Standard	35.7 MB	May 24, 2020, 2:34:56 PM by david.sarkies@internode.on.net	Tools
20200524t142332	Serving	12%	1	python27	Standard	35.7 MB	May 24, 2020, 2:24:23 PM by david.sarkies@internode.on.net	Tools
20200524t141153	Serving	12%	1	python27	Standard	35.7 MB	May 24, 2020, 2:12:50 PM by david.sarkies@internode.on.net	Tools
20200524t140117	Serving	12%	1	python27	Standard	35.7 MB	May 24, 2020, 2:02:14 PM by david.sarkies@internode.on.net	Tools
20200524t135013	Serving	12%	1	python27	Standard	37.4 MB	May 24, 2020, 1:51:13 PM by david.sarkies@internode.on.net	Tools
20200524t134711	Serving	16%	1	python27	Standard	37.4 MB	May 24, 2020, 1:48:05 PM by david.sarkies@internode.on.net	Tools
20200524t003648	Serving	0%	1	python27	Standard	37.4 MB	May 24, 2020, 12:37:57 AM by david.sarkies@internode.on.net	Tools

- 5) This is a list of the versions that are currently running. In the corner, between 'Migrate Traffic' and 'Show Info Panel' is an icon that looks like a split arrow. This is the traffic splitter. Select that and you will be diverted to the following screen:

←

Split traffic

You can split incoming traffic to different versions of your app. Traffic splitting is useful for slowly rolling out new versions or A/B testing different designs and features [Learn more](#)

**Split traffic by**

☒ IP address
 ☐ Cookie
 ☐ Random

**Traffic allocation**

20200524t134711	will receive the remaining	16 %	×
20200524t135013	<div></div>	12 %	×
20200524t140117	<div></div>	12 %	×
20200524t141153	<div></div>	12 %	×
20200524t142332	<div></div>	12 %	×
20200524t143405	<div></div>	12 %	×
20200524t144513	<div></div>	12 %	×
20200524t145809	<div></div>	12 %	×

+ Add version

Save

Cancel

- 6) Select either split traffic by IP Address, or random, and then select add version. This will give a list of versions to select. The versions are listed from the first deployed (at the top) to the last deployed. You will need to navigate down to the bottom to select those versions. The bar allows you to select how much traffic you want to divert to the selected versions.

# User Manual

After navigating to the site, you will be greeted by a Google Map that displays icons over places that have been visited by other users. You may have to zoom in on the map in order to view the icons. The icons will only appear on a zoom level of 15 or lower.

Also, due to the distributed nature of the app, you may end up in a location that is not Melbourne. You will note that San Francisco and New York do not have any locations mapped to it, though the other locations (Rome, Paris, London, Frankfurt, Hong Kong, and Melbourne) do.

## Creating An Account

If you wish to add your current location to the map or write reviews, you will first have to create an account. On the home page, select 'Sign Up' and fill in the form. Once signed up, an email will be sent to the address you signed up with. You will now be prompted to log in.

## Adding Your Location to the Map

To add your location to the map, you must first lock your location. Simply select the lock location button. This is used to ensure you cannot add your current location multiple times. After this, select "Add Place" and you will be taken to another webpage where you can fill in the information about the place. The form will change based on the type of place you are adding. The application currently supports user addition of Pubs/Bars, Cafes, Restaurants/Takeaway and Museums. Submit the form to add the place to the map.

## Writing and Viewing Reviews

To write a review, you must first navigate to the location on the map. Click on the icon and then the "View Place" button. You will then be taken to a page that displays the place's information. At the bottom of this page, there will be an "Add Review" button. Clicking this will take you to the review page, where you can fill in a short form to review a place. A rating is required, but review text is not. You will be limited to one review per place.

## References

- Ivasiuc, A 2016, 'Error importing Google Cloud Bigquery api module in python app', *Stack Overflow*, forum post, 30 November, viewed 11 May 2020,  
<<https://stackoverflow.com/questions/40886217/error-importing-google-cloud-bigquery-api-module-in-python-app>>
- Davila, M 2017, 'How to import BigQuery in AppEngine for Python', *Stack Overflow*, forum post, 29 March, viewed 11 May 2020,  
<<https://stackoverflow.com/questions/43085047/how-to-import-bigquery-in-appengine-for-python>>
- Steve, 2012, 'Google App Engine: Getting Sessions working with Python 2.7', *Stack Overflow*, forum post, October 4, viewed 4 May 2020,  
<<https://stackoverflow.com/questions/12737008/google-app-engine-getting-sessions-working-with-python-2-7/12737074#12737074>>
- Molina, A 2018, *Hashing Passwords in Python*, Vitosh Academy, viewed 28 April 2020,  
<<https://www.vitoshacademy.com/hashing-passwords-in-python/>>
- WebApp2 Authors 2011, *WebApp2 Documentation*, WebApp2 Authors, viewed 13 April 2020,  
<<https://webapp2.readthedocs.io/en/latest/#api-reference-webapp2>>
- Google, n.d., *BigQuery Client Libraries*, Google, viewed 11 May 2020,  
<<https://cloud.google.com/bigquery/docs/reference/libraries>>
- Google, n.d., *The Python 2 NDB Client Library Overview*, viewed 21 April 2020,  
<<https://cloud.google.com/appengine/docs/standard/python/ndb/>>
- 'Dynamically create a HTML form with Javascript', *Stack Overflow*, forum post, 21 July 2010, viewed 4 May 2020,  
<<https://stackoverflow.com/questions/3297143/dynamically-create-a-html-form-with-javascript>>
- 'Google map API v3 - Add multiple infowindows', *Stack Overflow*, forum post, 3 May 2015,  
<<https://stackoverflow.com/questions/30012913/google-map-api-v3-add-multiple-infowindows>>
- 'Google Maps API add marker with form, but getElementById only after second click', *Stack Overflow*, forum post, 29 May 2016  
<<https://stackoverflow.com/questions/37511911/google-maps-api-add-marker-with-form-but-getelementbyid-only-after-second-click>>
- 'Button Post HTML', *Stack Overflow*, forum post, 20 January 2011  
<<https://stackoverflow.com/questions/4666328/button-post-html>>

‘Use Jinja2 template engine in external javascript file’. *Stack Overflow*, forum post, 22 Feb 2014

<<https://stackoverflow.com/questions/21956500/use-jinja2-template-engine-in-external-javascript-file>>

*Flat Icons*, various authors (icons which are used in the maps)

<<https://www.flaticon.com/>>