



*Deploying a Python Flask Web Application
(extracts Text from the image and convert
into audio) to App Engine Flexible*

Submitted by

Aakash B (12203004) II year B.Tech CSE

M S Sucharita (121004248) III year B.Tech ECE

Leela Soundarya Y (121004145) III year B.Tech ECE

Abstract:

This Project comes up with an idea of extracting the text from the image and also converting it into audio. Image Text is the text information embedded or written in image of different form. Image text can be found in captured images, scanned documents, magazines, newspapers, posters etc. The information from these image documents would give higher efficiency and ease of access if it is converted to text form. The process by which Image Text converted into plain text is Text Extraction. Text Extraction is useful in information retrieving, searching, editing, documenting, archiving or reporting of image text. Conversion of the text extracted into audio is intended for people who are blind or have low vision, audio description has proven to be beneficial for a much broader audience. Information can be best intended through sound. This project uses the Vision API, Google Text-to-Speech API provided by Google Cloud Platform and Web application that performs the above extraction and audio file conversion is deployed in Google Cloud Platform(GCP).

Related Works:

1. Extracting Text from Image Document and Displaying Its Related Information by K.N. Natei*, J. Viradiya**, S. Sasikumar

This paper gives the idea about the extraction of text from Image. In this paper, text extraction from image documents, using combination of the two powerful methods Connected Component and Edge Based Method, in order to enhance performance and accuracy of text extraction.

2.Vision API Documentation

<https://cloud.google.com/vision/docs>

This documentation helps to understand deeply about the how to extract the text from the images using python code and hence how to integrate this with the Google Cloud Platform from the Web Application point of view.

3. Google Text-to-Speech Documentation

<https://cloud.google.com/text-to-speech/docs>

This documentation provided by Google helps us to deeply understand about the conversion of a given text into a audio file which enhance the vision of our project.

4. Faster Text-to-Speeches: Enhancing Blind People's Information Scanning with Faster Concurrent Speech

This paper gives the vision for our project. It makes us how we can help the blind people problem with Google Cloud Platform extensively. In this paper they text are converted into audio file which makes the blind people respond effectively.

Proposed architecture & services used:

This project is our new approach in which we have extensively used Web application to perform the text extraction from the image and convert the text into audio file. Then, the complete Web application is deployed in the App Engine, a serverless deployment in Google Cloud Platform (GCP).

For example, application allows a user to upload a photo of a handwritten image and gives the text as the output along with the audio file as output where he can listen to it or download the audio file.

The proposed architecture in order to perform this project are follows :-



Hence the above flow chart is performed by using the following Google Cloud Platform(GCP) resources

1.Google Service Accounts: Service Account to access the Google Cloud APIs when testing locally. And once services account is created the appropriate permissions are given to it. And finally service Account key is created.

2.Google App Engine: With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go. Hence our entire web application is uploaded into the App Engine. Now we can perform.

3.Virtualenv: Though it is not Google Resources we need to create a virtual environment for running the python-3. Since we used Flask web framework it is necessary for the use of virtual environment.

4.Cloud Storage Bucket: Cloud Storage Bucket is very necessary resource because whatever the images that are

uploaded are stored in the cloud storage bucket. We need to store the audio file into our bucket for future use. In our project, Cloud Bucket name is same as Project Id of the lab

5.Vision API: It is the most important API that extracts the text from the image. This API is free to use and thereby helps us the extract the text easily.

6.Cloud Text-to-Speech API: This API is used to convert the text to audio. A request is made by passing the text as a parameter and the output is provided by the API as a audio file.

7.Cloud Monitoring: This resource helps us to monitor the https traffic from the end user who uses our web application from the host website. It also ensures the plot between the CPU.

Implementation:

To perform this project, we have to use one qwiklabs lab

‘Deploying a Python Flask Web Application to App Engine Flexible’

Now we need to enable Cloud Text-to-Speech API. Go to the console and select API & Services and search text to speech and enable the API because we are not provided with Cloud Text-to-Speech API by default in this lab.

We use Google Cloud Shell throughout this Lab because it is easy to perform certain installations. Firstly, we are going to work with our current project id

For that type the following command

```
gcloud config set project [project_id]
```

Once we are sure that we are working with our current project then we can move further. In Cloud Shell on the command-line, run the following command to clone the GitHub repository:(The following GitHub has the web application code)

```
git clone https://github.com/itsmeaakash77/GCP-PROJECT.git
```

and change the directory to GCP-Project using

```
cd GCP-PROJECT
```

Now we have to authenticate API Requests:-

Set an environment variable for [YOUR_PROJECT_ID], replacing [YOUR_PROJECT_ID] with our project ID:

```
export YOUR_PROJECT_ID = [YOUR_PROJECT_ID]
```

Create a Service Account to access the Google Cloud APIs when testing locally:

```
gcloud iam service-accounts create qwiklab \ --display-name  
"My Qwiklab Service Account"
```





Give your newly created Service Account appropriate permissions:

```
gcloud projects add-iam-policy-binding ${PROJECT_ID} \  
--member  
ServiceAccount:qwiklab@{PROJECT_ID}.iam.gserviceaccount.  
com\ --role roles/owner
```

After creating a Service Account, create a Service Account key:

```
gcloud iam service-accounts keys create ~/key.json \
--iam-account
qwiklab@${PROJECT_ID}.iam.gserviceaccount.com
```

This command generates a service account key stored in a JSON file named key.json in our home directory. This key helps us to access the API with ease.

Service accounts + CREATE SERVICE ACCOUNT DELETE SHOW INFO PANEL							
Service accounts for project "qwiklabs-gcp-00-8f0ee519e187"							
A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more							
Filter table							
<input type="checkbox"/>	Email	Status	Name ↑	Description	Key ID	Key creation date	Actions
<input type="checkbox"/>	 qwiklabs-gcp-00-8f0ee519e187@appspot.gserviceaccount.com	✓	App Engine default service account		No keys		⋮
<input type="checkbox"/>	 924010349986-compute@developer.gserviceaccount.com	✓	Compute Engine default service account		No keys		⋮
<input type="checkbox"/>	 qwiklab@qwiklabs-gcp-00-8f0ee519e187.iam.gserviceaccount.com	✓	My Qwiklab Service Account		4924f5bf1c7c31b909b499f72645751e9b0a1473	Mar 28, 2020	⋮
<input type="checkbox"/>	 qwiklabs-gcp-00-8f0ee519e187@qwiklabs-gcp-00-8f0ee519e187.iam.gserviceaccount.com	✓	Qwiklabs User Service Account		c4ef43b1952d4546fef45951bc8a0a50df71c7aa	Mar 25, 2020	⋮

Using the absolute path of the generated key, we set an environment variable for our service account key:

```
export
GOOGLE_APPLICATION_CREDENTIALS="/home/${USER}/key.json"
```

Test Applications Locally

Create an isolated Python 3 environment named env with [virtualenv](#) using following command

```
virtualenv -p python3 env
```

Now entering into newly created virtualenv named env:

```
source env/bin/activate
```

Use pip to install dependencies for your project from the requirements.txt file:

```
pip install -r requirements.txt
```

The requirements.txt file is a list of package dependencies you need for your project. The above command downloaded all of these listed package dependencies to the virtualenv

Now we need to create the App Engine app

```
gcloud app create
```

[and for our lab we enter 13 to Select a Region that supports App Engine Flexible for Python]

Now create the Storage Bucket

Now, we set the environment variable CLOUD_STORAGE_BUCKET equal to the name of PROJECT_ID. (It is generally recommended to name bucket the same as your PROJECT_ID for convenience purposes).

```
export CLOUD_STORAGE_BUCKET=${PROJECT_ID}
```

Now we run the following command to create a bucket with the same name as your PROJECT_ID.

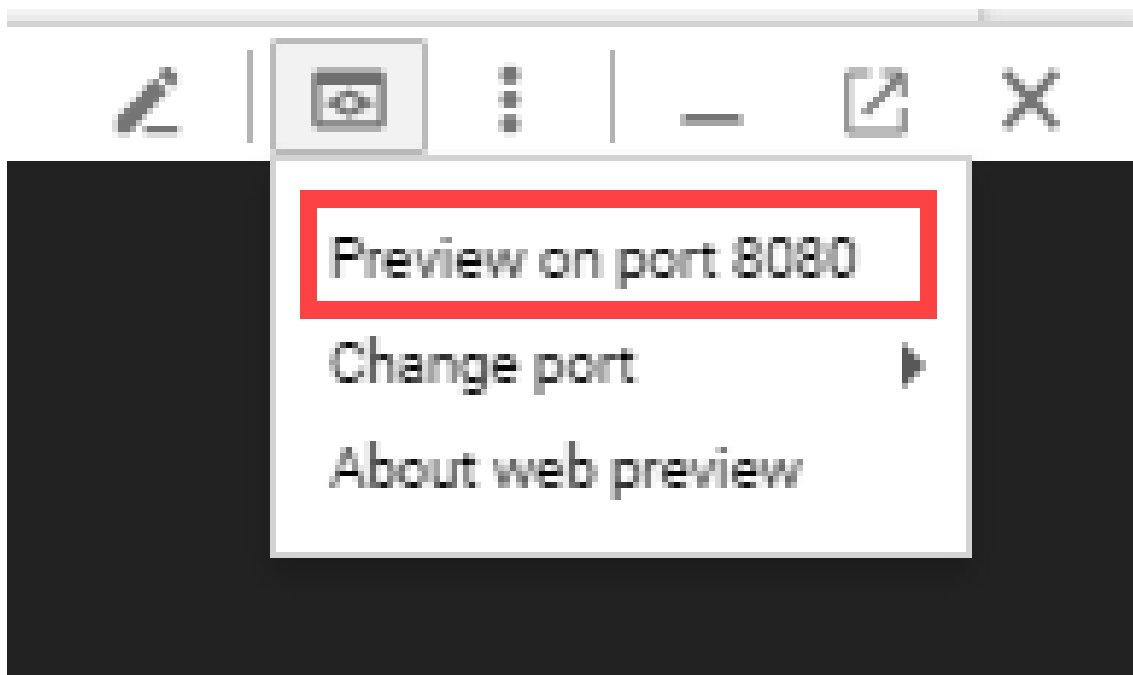

```
gsutil mb gs://${PROJECT_ID}
```

Execute the following command to start application:

```
python main.py
```

```
127.0.0.1 - - [28/Mar/2020 15:19:27] "GET / HTTP/1.1" 200 -  
(env) student_00_9e874bd36c76@cloudshell:~/GCP-PROJECT (qwiklabs-gcp-00-8f0ee519e187)$ python main.py  
* Serving Flask app "main" (lazy loading)  
* Environment: production  
  WARNING: This is a development server. Do not use it in a production deployment.  
  Use a production WSGI server instead.  
* Debug mode: on  
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 203-496-098  
127.0.0.1 - - [28/Mar/2020 15:27:04] "GET /?authuser=0 HTTP/1.1" 200 -  
█
```

Once the application starts, click on the Web Preview icon in the Cloud Shell toolbar and choose "Preview on port 8080."



A tab is opened in a browser and it gets connected to the server that just started. The screen will appear like this.

Google Cloud Platform - Text - extraction from images

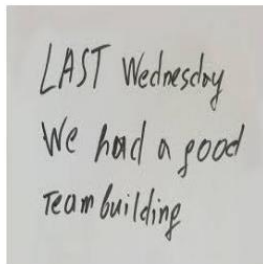
Submitted by

B.Aakash (12203004) II year B. Tech CSE

Y. Leela Soundarya(121004145) III year B.Tech ECE

M S Sucharita(121004248) III year B.Tech ECE

Upload File: No file chosen



Choose file button is the one where we uploaded our image contains the text/number. And after finishing this verify the Cloud Bucket storage. It should be like this

← Bucket details									
qwiklabs-gcp-01-19be8713dff1									
OBJECTS CONFIGURATION PERMISSIONS RETENTION LIFECYCLE									
Buckets > quiklabs-gcp-01-19be8713dff1									
UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DELETE									
Filter by object or folder name prefix									
<input type="checkbox"/>	Name	Type	Storage class	Last modified	Public access	Encryption	Retention expiration date	Holds	
<input type="checkbox"/>	download.jfif	image/jpeg	Standard	Apr 2, 202...	Public to internet	Copy URL	Google-managed key	—	None
<input type="checkbox"/>	images.jfif	image/jpeg	Standard	Apr 2, 202...	Public to internet	Copy URL	Google-managed key	—	None
<input type="checkbox"/>	output.mp3	application/octet-stream	Standard	Apr 2, 202...	Public to internet	Copy URL	Google-managed key	—	None
<input type="checkbox"/>	output.wav	application/octet-stream	Standard	Apr 2, 202...	Public to internet	Copy URL	Google-managed key	—	None

And finally now we are going to deploy the app to App Engine Flexible once tested successfully.

App Engine Flexible uses a file called app.yaml to describe an application's deployment configuration. If this file is not

present, App Engine will try to guess the deployment configuration. However, it is a good idea to provide this file.

Next, we will modify app.yaml using an editor of our nano editor.

```
nano app.yaml
```

replace <your-cloud-storage-bucket> with the name of our Cloud Storage bucket

then click **ctrl+x** and press 'Y' and hit enter to return back to the shell.

Deploy your app on App Engine by using gcloud:

```
gcloud app deploy
```

```
^C(env) student_00_9e874bd36c76@cloudshell:~/GCP-PROJECT (qwiklabs-gcp-00-8f0ee519e187)$ nano app.yaml
(env) student_00_9e874bd36c76@cloudshell:~/GCP-PROJECT (qwiklabs-gcp-00-8f0ee519e187)$ gcloud app deploy
```

After the application is deployed, the url looks something like this.

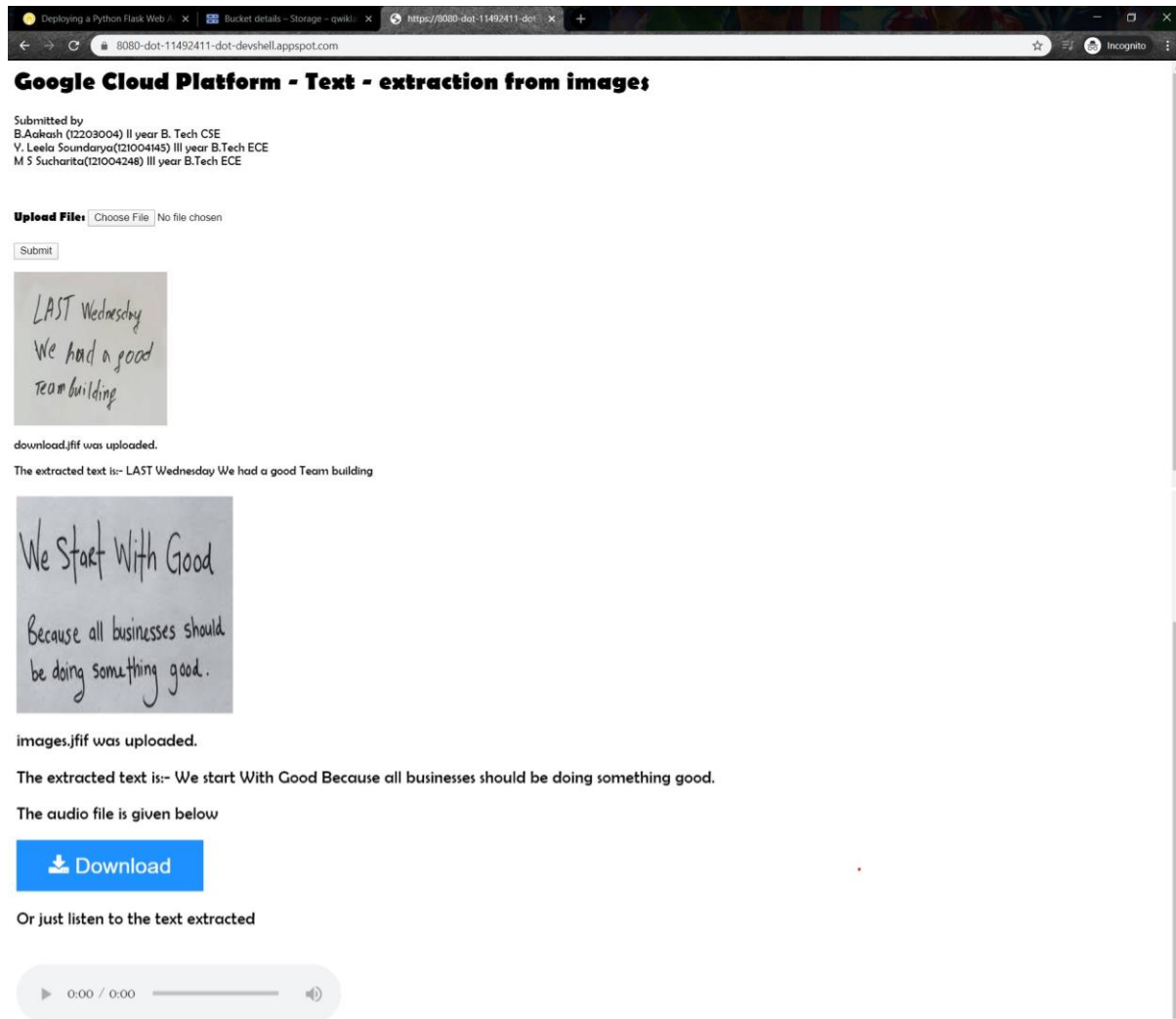
```
https://<PROJECT_ID>.appspot.com
```

NOTE: Being done in qwiklabs, this website is active till the qwiklabs session is active, if it expires then the user cannot see this website working.

Results and Discussions:

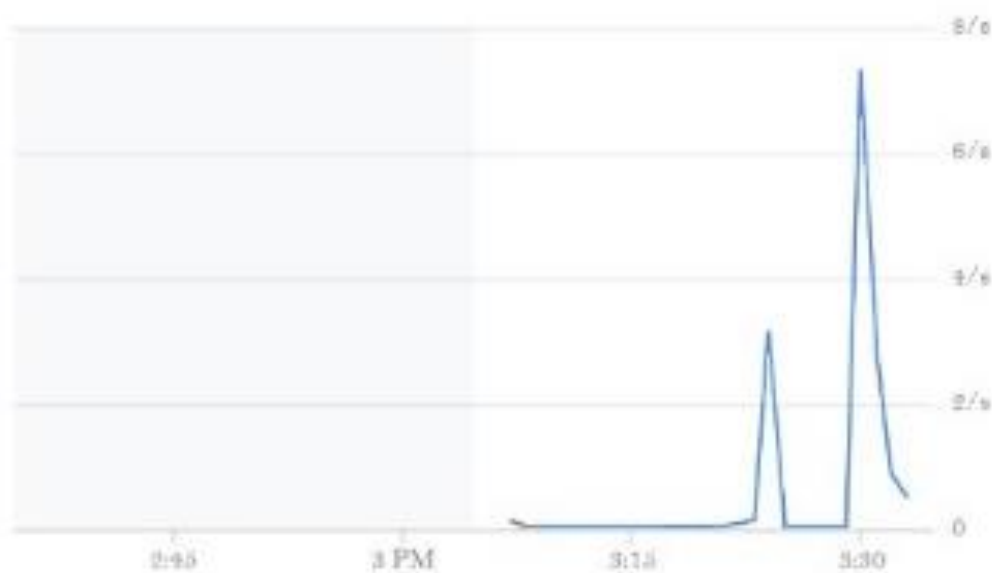
This is how the website looks at that particular

“https://****.appspot.com”



The following screenshots of the qwiklabs are the results found in our project which are found in the API dashboard.

Traffic



Errors



Median latency



Conclusion and Future Works:

Hence we conclude that with the help of Google Cloud we have deployed our Web Applications using server-less deployment using App Engine in Google Cloud Platform(GCP). As a result of our project we have extracted the text from the image and the text is converted as the audio file.

In future, we can introduce the use of online images for uploading the image file. And also, we can enhance the text-extraction in which includes the combination of different languages and which includes some strike text. Thereby, we can enhance the perfect text-extraction from the image. We can include the Google Translate for the different language text to user language.