Task 1: Part A – Compute Engine Creation

- **Setting-up Assignment Configuration**

Created a new project on google cloud “cloudassign-382706”

Step 1: Program Setup and account initialization

Command: *gcloud init*

In this step, I have opted for default reconfiguration setting, moreover, I have selected my email ID associated with official account. All the necessary steps have been performed through that. It can be visualized in the steps below.

Google Cloud SDK Shell - gcloud init

```
Please enter your numeric choice:  1

Your current configuration has been set to: [default]

You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

Choose the account you would like to use to perform operations for this configuration:
 [1] shahhussaink280@gmail.com
 [2] Log in with a new account
Please enter your numeric choice:  1

You are logged in as: [shahhussaink280@gmail.com].
```

Step 2: Next to select assignment:

In this step, I have selected my project created in google cloud

```
Pick cloud project to use:
 [1] alert-function-377915
 [2] beaming-prism-377916
 [3] cloudasignment
 [4] cohesive-geode-377915
 [5] crypto-symbol-377916
 [6] sincere-octane-377916
 [7] vocal-vigil-377916
 [8] Enter a project ID
 [9] Create a new project
Please enter numeric choice or text value (must exactly match list item):  3

Your current project has been set to: [cloudasignment].
```

Step 3: Adjust Time Zone

Here, I am selected an optimal time zone for my assignment. Figure below shows more about it.

```
Your current project has been set to: [cloudasignment].

Do you want to configure a default Compute Region and Zone? (Y/n)?  Y

Which Google Compute Engine zone would you like to use as project default?
If you do not specify a zone via a command line flag while working with Compute Engine resources, the default is assumed.
 [1] us-east1-b
 [2] us-east1-c
 [3] us-east1-d
 [4] us-east4-c
 [5] us-east4-b
 [6] us-east4-a
 [7] us-central1-c
 [8] us-central1-a
 [9] us-central1-f
 [10] us-central1-b
 [11] us-west1-b
 [12] us-west1-c
 [13] us-west1-a
 [14] europe-west4-a
 [15] europe-west4-b
 [16] europe-west4-c
 [17] europe-west1-b
 [18] europe-west1-d
 [19] europe-west1-c
 [20] europe-west3-c
 [21] europe-west3-a
 [22] europe-west3-b
 [23] europe-west2-c
 [24] europe-west2-b
 [25] europe-west2-a
 [26] asia-east1-b
 [27] asia-east1-a
 [28] asia-east1-c
 [29] asia-southeast1-b
 [30] asia-southeast1-a
 [31] asia-southeast1-c
 [32] asia-northeast1-b
 [33] asia-northeast1-c
 [34] asia-northeast1-a
 [35] asia-south1-c
 [36] asia-south1-b
 [37] asia-south1-a
 [38] australia-southeast1-b
 [39] australia-southeast1-c
 [40] australia-southeast1-a
 [41] southamerica-east1-b
 [42] southamerica-east1-c
 [43] southamerica-east1-a
 [44] asia-east2-a
 [45] asia-east2-b
 [46] asia-east2-c
 [47] asia-northeast2-a
 [48] asia-northeast2-b
 [49] asia-northeast2-c
 [50] asia-northeast3-a
Did not print [63] options.
Too many options [113]. Enter "list" at prompt to print choices fully.
```

```
Your Google Cloud SDK is configured and ready to use!

* Commands that require authentication will use shahhussaink280@gmail.com by default
* Commands will reference project `cloudasignment` by default
* Compute Engine commands will use region `us-central1` by default
* Compute Engine commands will use zone `us-central1-a` by default

Run `gcloud help config` to learn how to change individual settings

This gcloud configuration is called [default]. You can create additional configurations if you work with multiple accounts and/or projects.
Run `gcloud topic configurations` to learn more.

Some things to try next:

* Run `gcloud --help` to see the Cloud Platform services you can interact with. And run `gcloud help COMMAND` to get help on any gcloud command.
* Run `gcloud topic --help` to learn about advanced features of the SDK like arg files and output formatting
* Run `gcloud cheat-sheet` to see a roster of go-to `gcloud` commands.

C:\Program Files (x86)\Google\Cloud SDK>
```

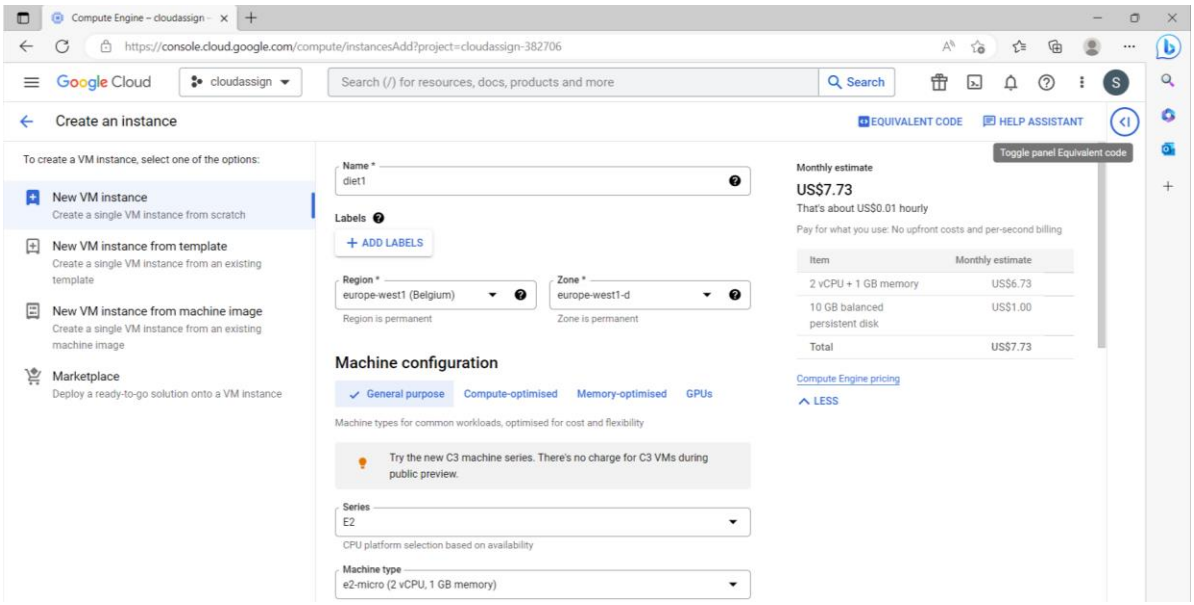
In next, I am going to execute the part a of task 1. Which is based on creating of virtual machine, I have created an instance and connected it to my account.

Part a:

Instance Name: diet1

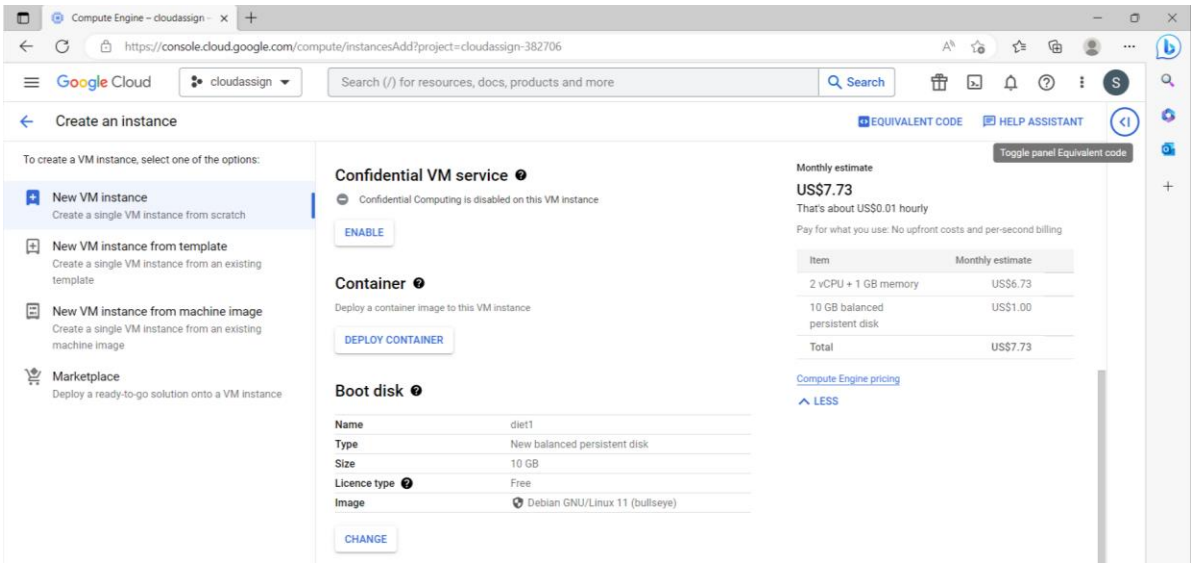
Zone: Europe-west1-d [As per my understanding it is the economical option to be considered]

Machine Type: e2-micor

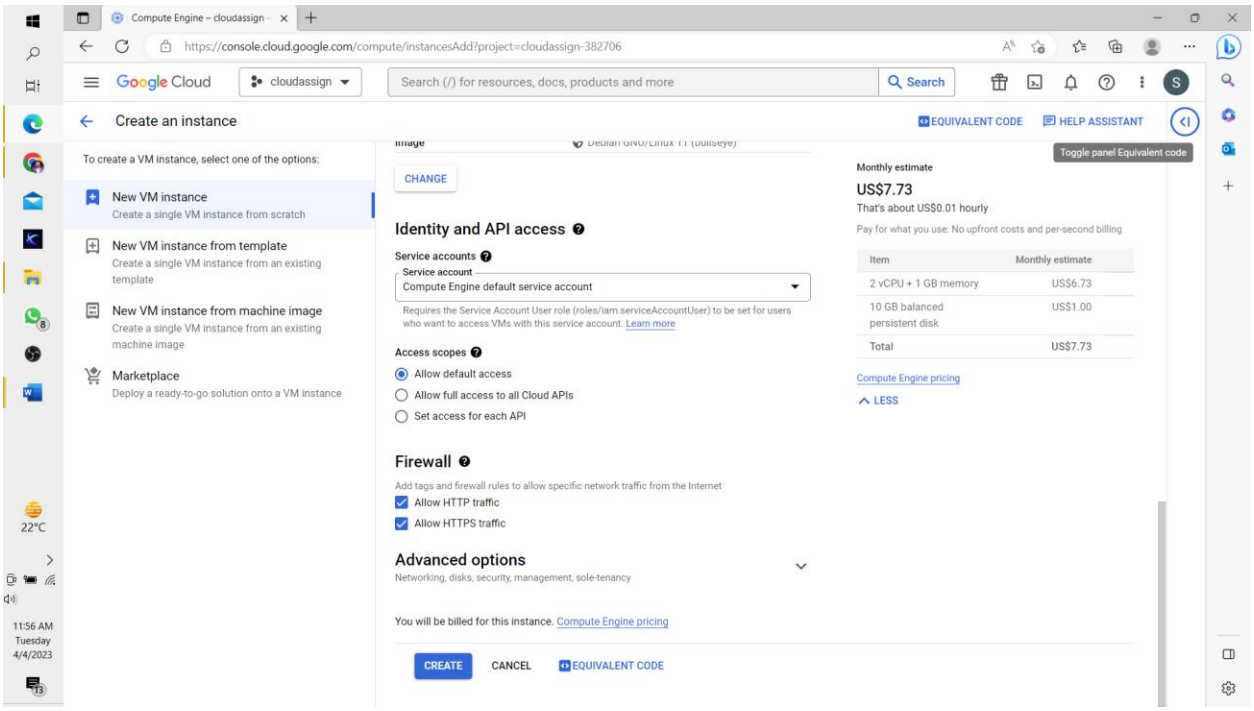


Boot Disk Size: 10 GB

More details can be observed below



As the requirement is based to include HTTP and HTTPs traffic, so I have opted both tags and configured firewall. Moreover default API access has been opted.

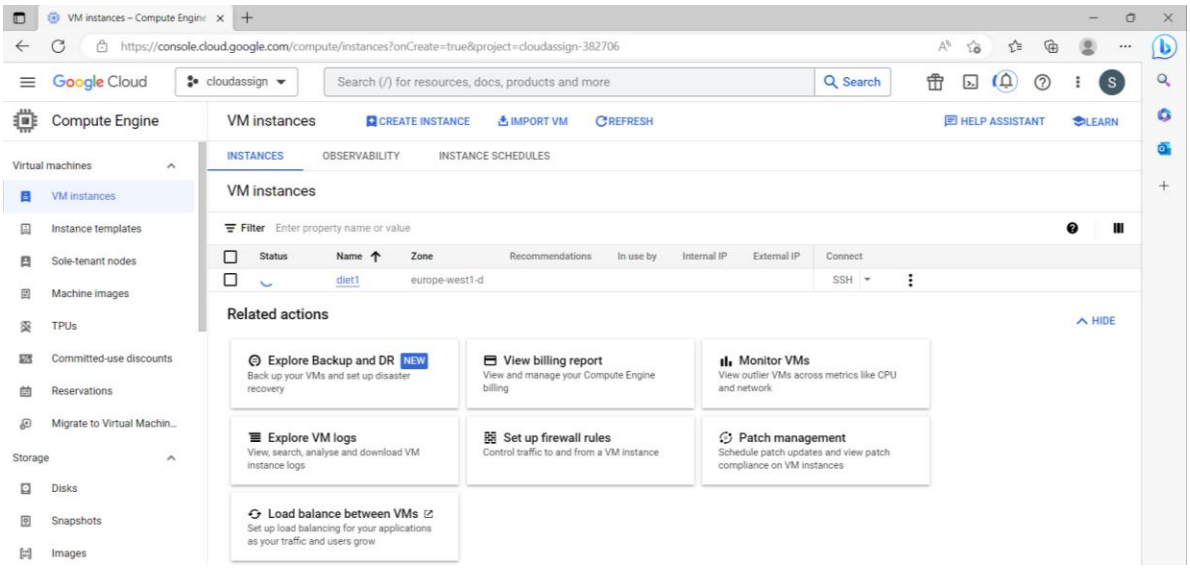


- **Code for above instance and virtual machine adjustments:**

```
gcloud compute instances create diet1 \
    --project=cloudassign-382706 \
    --zone=europe-west1-d \
    --machine-type=e2-micro \
    --network-interface=network-tier=PREMIUM,subnet=default \
    --maintenance-policy=MIGRATE \
    --provisioning-model=STANDARD \
    --service-account=545786750704-compute@developer.gserviceaccount.com \
    --
scopes=https://www.googleapis.com/auth/devstorage.read_only,https://www.googleapis.com
/auth/logging.write,https://www.googleapis.com/auth/monitoring.write,https://www.googl
eapis.com/auth/servicecontrol,https://www.googleapis.com/auth/service.management.reado
nly,https://www.googleapis.com/auth/trace.append \
    --tags=http-server,https-server \
    --create-disk=auto-delete=yes,boot=yes,device-name=diet1,image=projects/debian-
cloud/global/images/debian-11-bullseye-
v20230306,mode=rw,size=10,type=projects/cloudassign-382706/zones/ europe-west1-d
/diskTypes/pd-balanced \
    --no-shielded-secure-boot \
    --shielded-vtpm \
    --shielded-integrity-monitoring \
    --labels=ec-src=vm_add-gcloud \
    --reservation-affinity=any
```

Now, Virtual Machine has been configured which can be seen in the google cloud [Screen Shot Attached on Next Page]

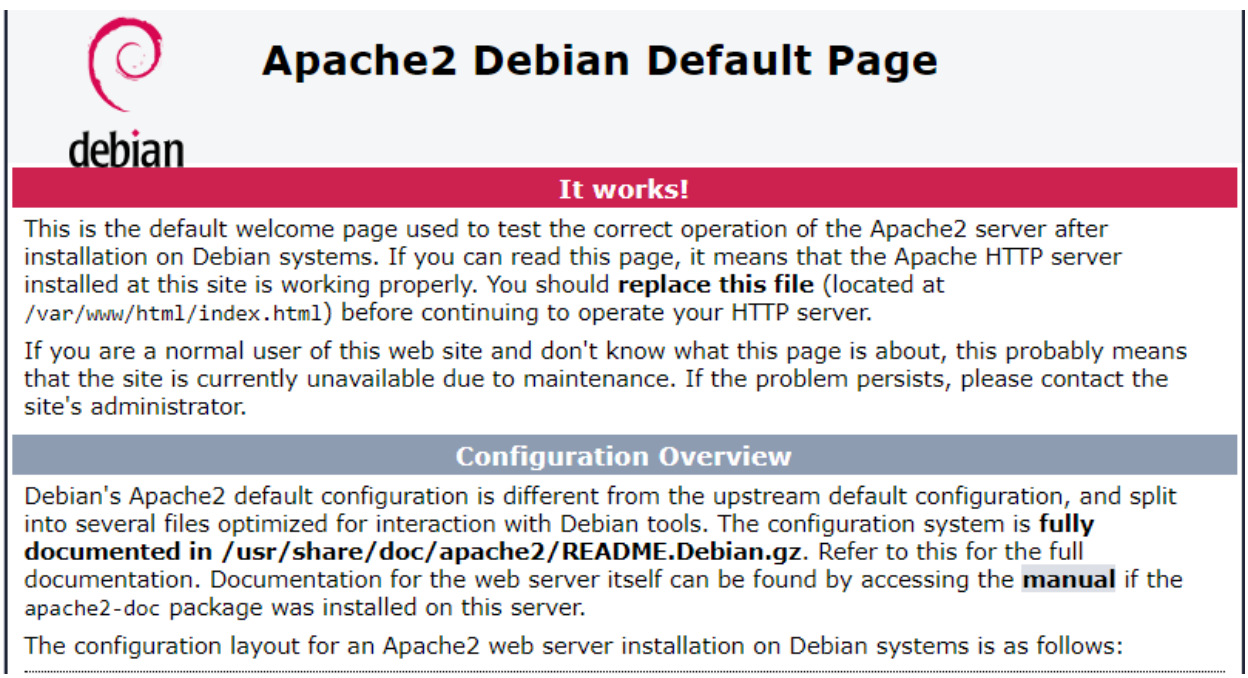
Reference: https://www.youtube.com/watch?v=pHOjFXTsLCg



Part b: Installation of Web Server

Command: gcloud compute ssh diet1 --zone=europe-west1-d --command='sudo apt-get update && sudo apt-get install -y apache2'

Apache 2 is a popular open-source web server software developed by the Apache Software Foundation. It is widely used to serve static and dynamic content on the World Wide Web and supports multiple operating systems and programming languages. Apache 2 is known for its robustness, scalability, and flexibility, making it a top choice for web developers and administrators.



Part C:
Step 1: Copied a photographic image and showed in through a URL

Command: gcloud compute scp C:/Users/shah1/pictures/picture1.jpg diet1:/var/www/html

```
C:/Users/shah1>gcloud compute scp C:/Users/shah1/pictures/picture1.jpg diet1:/var/www/html
```

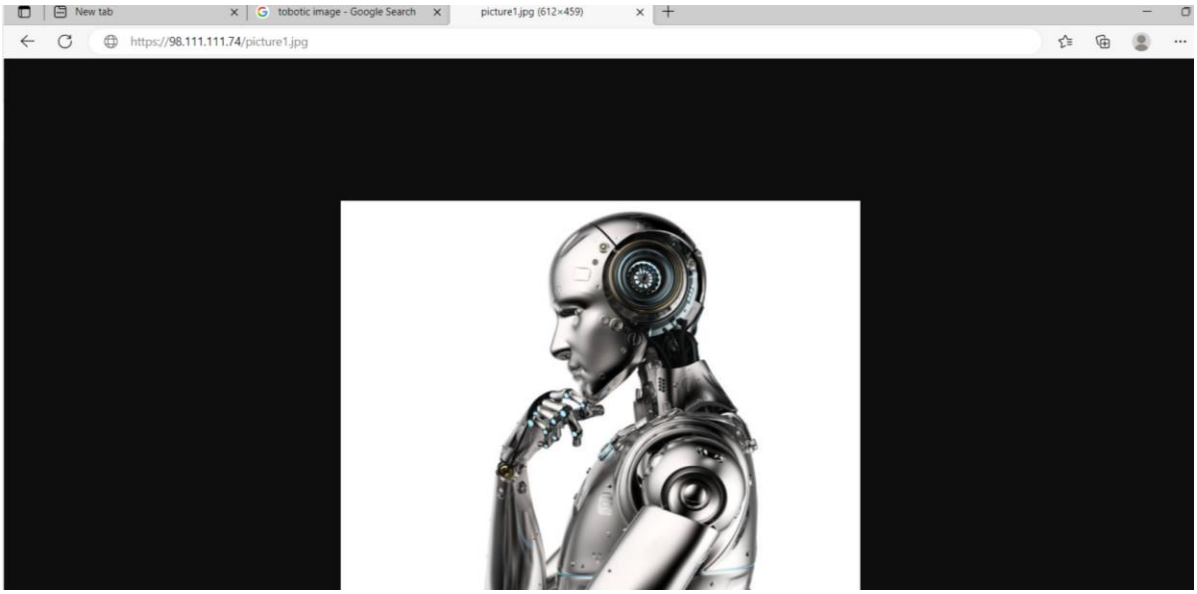
Output

```
| 194 kB | 194.2 kB/s | ETA: 00:00:00 | 100%
```

Here it can be observed that the image has been successfully uploaded. I have uploaded an image of a robot. Lets see how it looks on the server.
Step 2: Server- restarted apache2

Command: gcloud compute ssh cw- diet1--zone=europe-west1-d --command='sudo service apache2 restart'

Step 3: Showing Imapge using my URL: *98.111.111.74*



Part D: Development and testing of App Engine:
Preferred language: JavaScript
App Engine: Local
Code:

```
const http = require('http');
const hostname = '0.0.0.0';
const port = process.env.PORT || 8080;
const server = http.createServer((req, res) => {
res.statusCode = 200;
res.setHeader('Content-Type', 'text/plain');
res.end('Name: Shah Hussain and My student Number: S22*****\n');});
server.listen(port, hostname, () => {
console.log(`Server running at http://${hostname}:${port}/`);});
```

```
File Edit Selection View Go Run Terminal Help                    • AppEngine.js - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features.  Manage  Learn More
AppEngine.js ●   NLP Analysis of Project Gutenberg Part 1.ipynb

1  const http = require('http');
2  const hostname = '0.0.0.0';
3  const port = process.env.PORT || 8080;
4  const server = http.createServer((req, res) => {
5  res.statusCode = 200;
6  res.setHeader('Content-Type', 'text/plain');
7  res.end('Name: Shah Hussain and My student Number: S22*****\\n');});
8  server.listen(port, hostname, () => {
9  console.log(`Server running at http://${hostname}:${port}/`);});
10
```

Now Run the Epp Engine through powershell;

Commands:

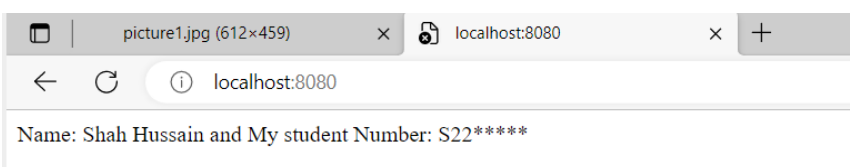
gcloud app create --region=europ-west1

gcloud app deploy

```
Services to deploy:

descriptor:                  [C:\Users\Shah1\documents\diet1-APP\app.yaml]
source:                      [C:\Users\Shah1\documents\diet1-APP-APP]
target project:              [cloudassicloudassign-382706 ]
target service:              [default]
target version:              [20230330t214842]
target url:                  [https://cloudassicloudassign-382706.lm.r.appspot.com]
target service account:      [App Engine default service account]


Do you want to continue (Y/n)?  y

Beginning deployment of service [default]...
#=============================================================#
#= Uploading 0 files to Google Cloud Storage                 =#
#=============================================================#
File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://mmi226816c1d1-381914.lm.r.appspot.com]

You can stream logs from the command line by running:
  $ gcloud app logs tail -s default

To view your application in the web browser run:
  $ gcloud app browse
```

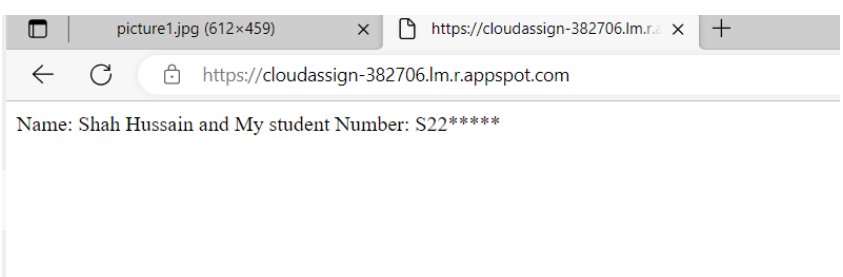
Now Starting app locally

Command: npm start



Now Starting app remotely

Link: https://cloudassign-382706.lm.r.appspot.com/



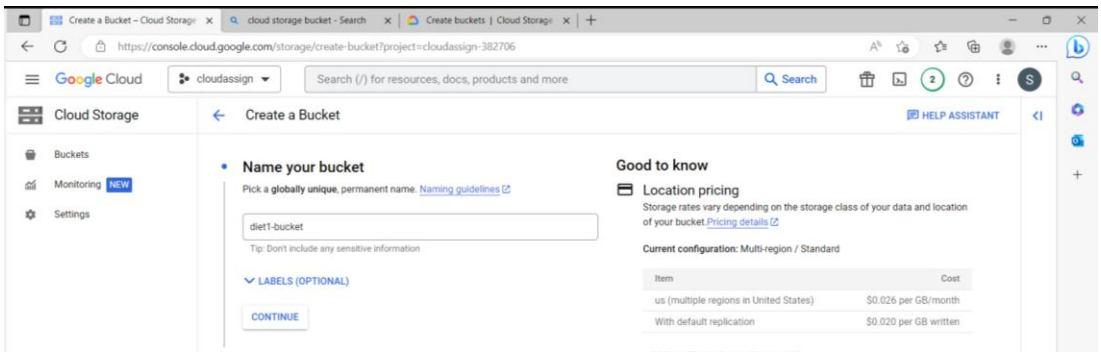
Task 2:

Part a: Creating bucket with two regions and all are publicly accessible

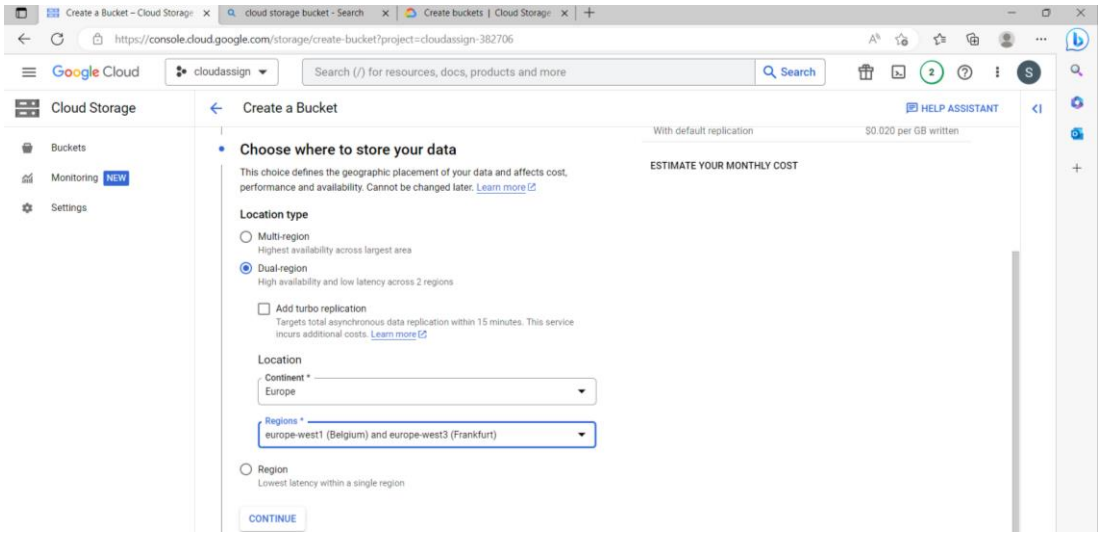
Google Cloud Storage buckets are cloud-based object storage that allows users to store and retrieve data from anywhere on the internet. These buckets can be configured as publicly accessible, which means that anyone with the bucket's URL can access the data stored in it. However, it's important to keep in mind the security implications of making buckets publicly accessible and to set appropriate permissions to ensure the safety of sensitive data.

Bucket Name: diet1-bucket

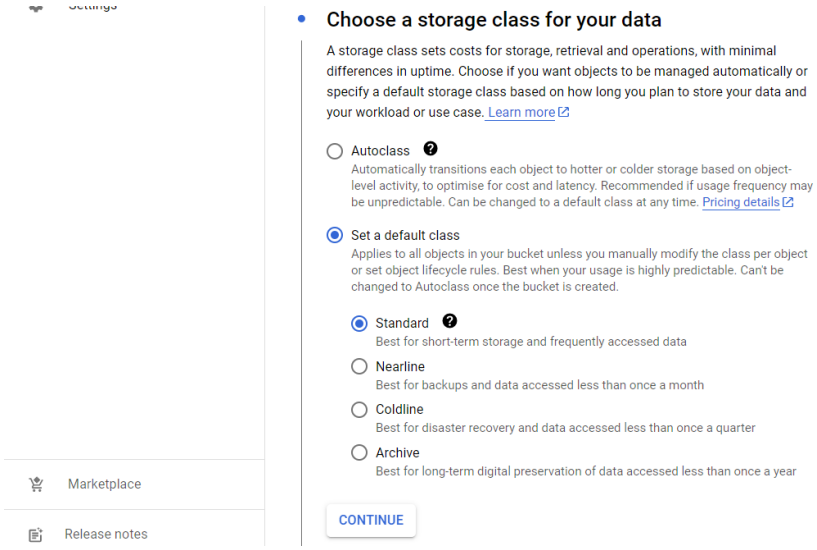
Step 1: Creating bucket Name



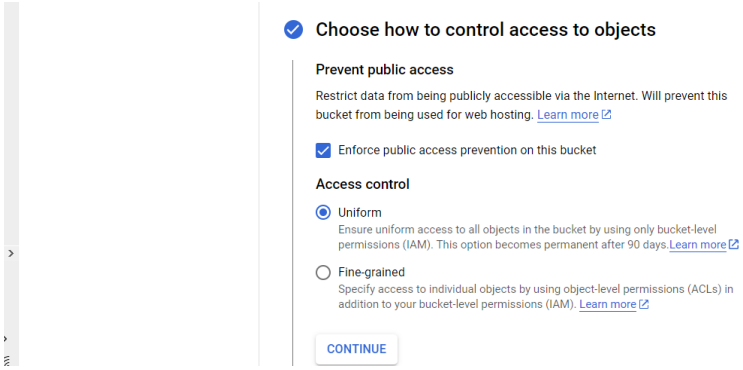
Step 2: Selected Europe continent with two regions.



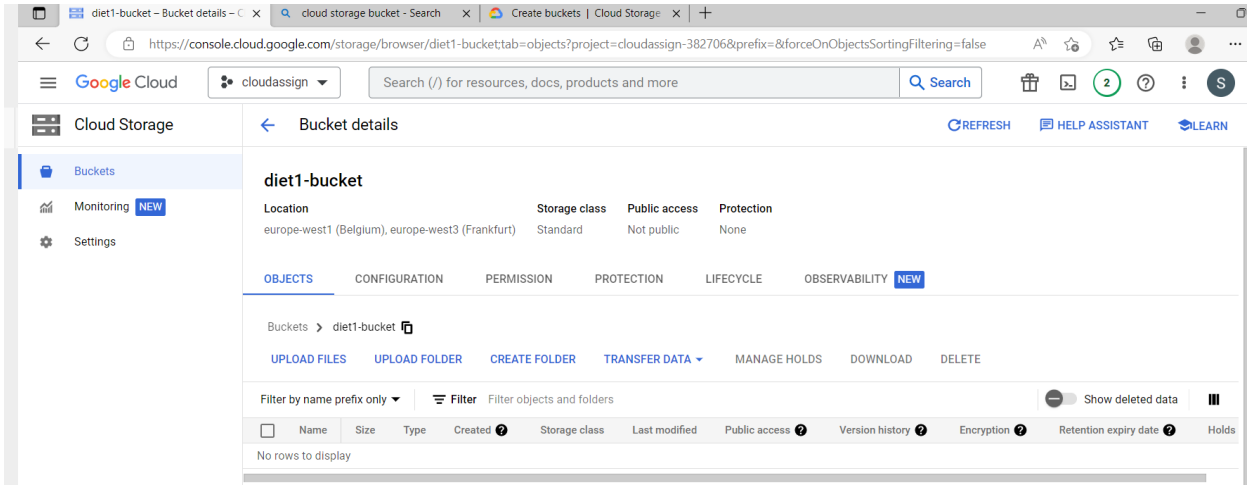
Step 3: Selected default class for storage of my data which is basically three pictures



Step 4: Applying uniform access control to bucket storage data



Finally, bucket, diet1-bucket has been created in the google cloud console with all required characteristics.



Part B

Uploading three random images to my bucket: diet1-bucket

Commands:

gsutil cp picture2.jpg gs://diet1-bucket/picture2.jpg

gsutil cp picture3.jpg gs://diet1-bucket/picture3.jpg

gsutil cp picture4.png gs://diet1-bucket/picture4.jpg

As I want to access these publicly, following commands are going to be implemented in shell

Commands:

gsutil acl ch -u AllUsers:R gs://diet1-bucket/picture2.jpg

gsutil acl ch -u AllUsers:R gs://diet1-bucket/picture3.jpg

gsutil acl ch -u AllUsers:R gs://diet1-bucket/picture4.jpg

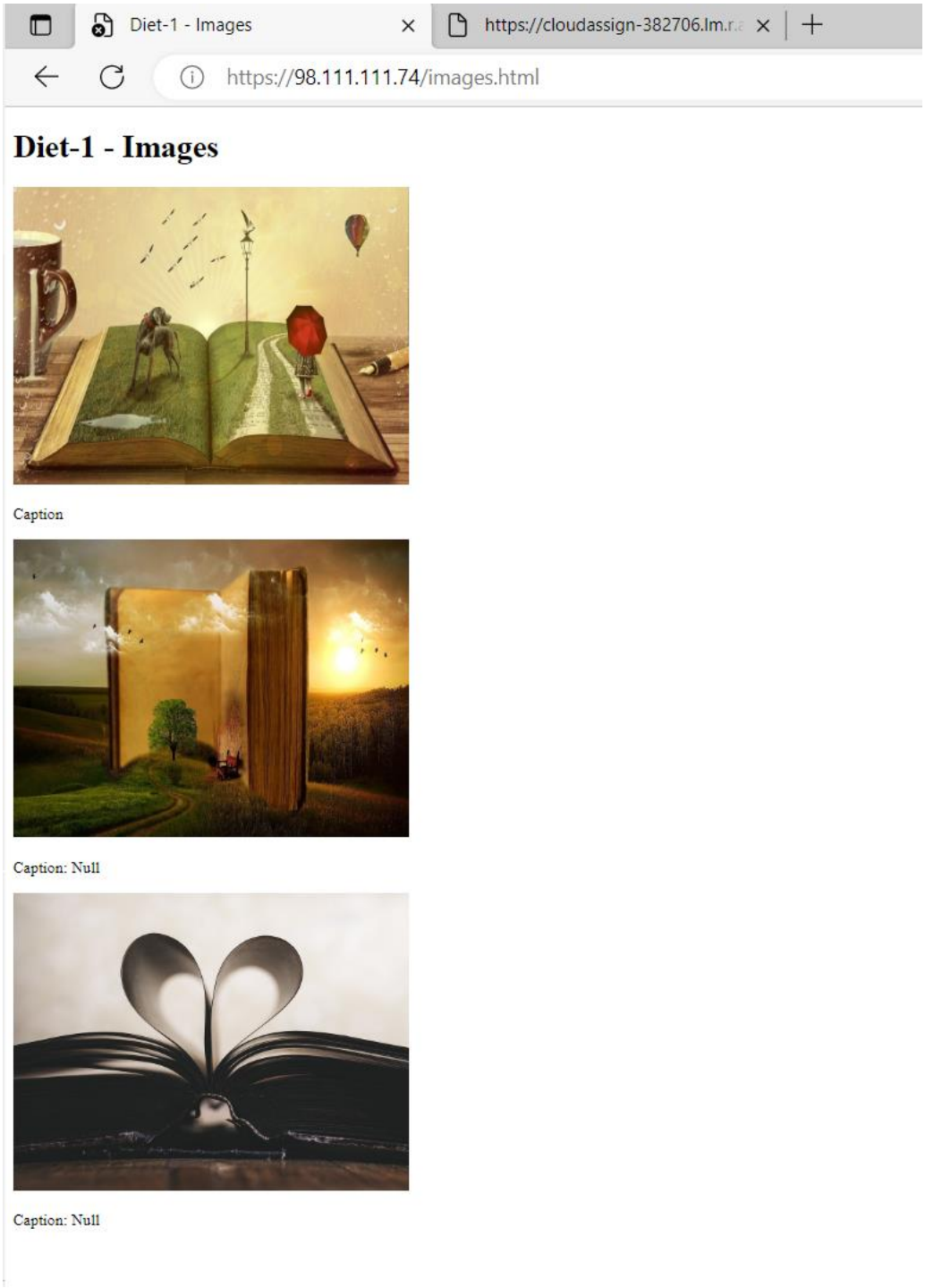
Part C: HTML Code:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Diet-1 - Images</title>
  </head>
  <body>
    <h1>Diet-1 - Images</h1> <div>
      <img
        src="https://storage.googleapis.com/diet1-bucket/picture2.jpg"
        alt="Picture 1" />
      <p>Caption</p>
    </div>
    <div>
      <img
        src="https://storage.googleapis.com/diet1-bucket/picture3.jpg"
        alt="Picture 2" />
      <p>Caption</p>
    </div>
    <div>
      <img
        src=" https://storage.googleapis.com/diet1-bucket/picture4.jpg "
        alt="Picture 3"/>
      <p>Caption</p> </div> </body></html>
```

Copied images.html file to apache2 default directory using following command

Command:
gcloud compute scp C:/Users/shah1/Documents/images.html diet1:/var/www/html

Lets view the uploaded images



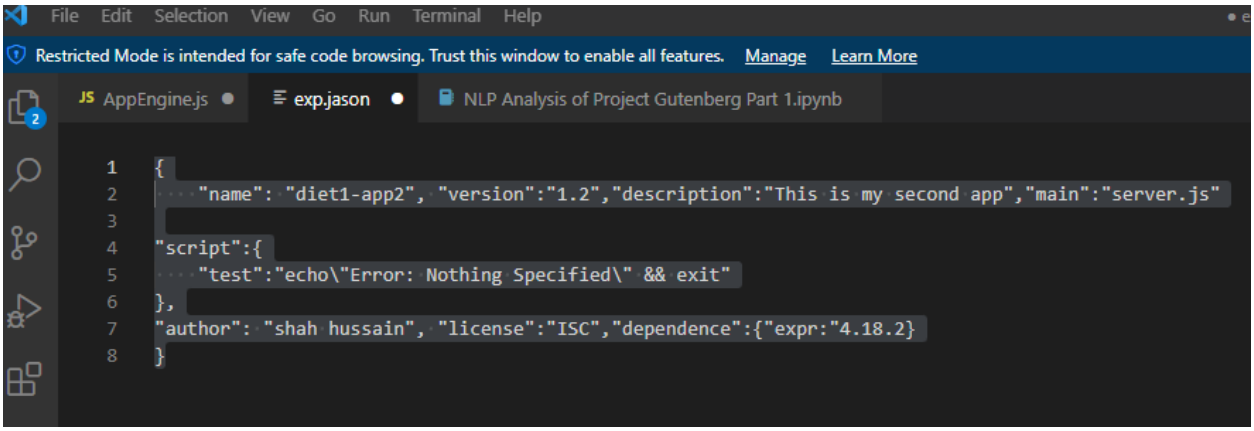
Part D: App Engine Development and testing

Command:

npm init

Code for App:

```
{
   "name": "diet1-app2", "version":"1.2","description":"This is my second app","main":"server.js"
"script":{
   "test":"echo\"Error: Nothing Specified\" && exit"
},
"author": "shah hussain", "license":"ISC","dependence":{"expr:"4.18.2}
}
```

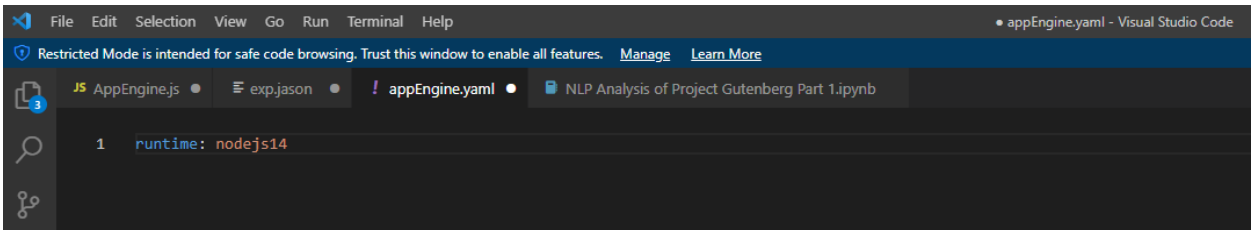


I have created code for app deployment. Lets Install express and runit.

Command:

npm install exp

AppEngine.yaml



- **HTML code for images.html**

As I have to observe the output, this html setting will be used to see the webpage.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Diet-1 - Images</title>
  </head>
  <body>
    <h1>
          Diet-1 – Images
    </h1>
    <div>
      <img src="https://pixabay.com/photos/paper-heart-symbol-romance-1100254/" alt="Picture 1">
      <p>
          Caption
     </p>
    </div>
    <div>
      <img src="https://pixabay.com/illustrations/book-old-surreal-fantasy-pages-863418/" alt="Picture 2">
      <p>
          Caption: Null
      </p>
    </div>
    <div>
      <img src="https://cdn.pixabay.com/photo/2015/12/19/20/32/paper-1100254_960_720.jpg"
alt="Picture 3">
      <p>Caption: Null</p>
    </div>
  </body></html>
```

- **Code for server.js**

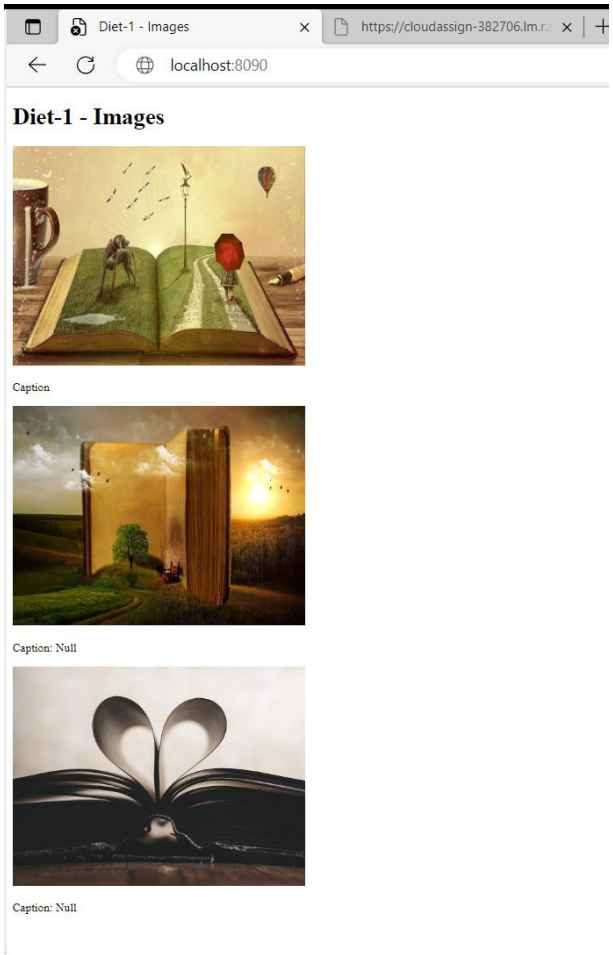
```
const express = require('express');
const app = express();
app.get('/images/:id', (req, res) => {
  const imageId = parseInt(req.params.id);
  const captions = {
1: 'Caption: ',
2: 'Caption: Null',
 3: 'Caption: Null'
};
  const caption = captions[imageId] || 'Image Not Found';
  const html = `<html>
              <head>
                <title>Image ${imageId}</title>
              </head>
              <body>
                <h1>Image ${imageId}</h1>
                <div>
                  <img src="https://storage.googleapis.com/diet1-bucket/image${imageId}.png" alt="Picture
1">
                  <p>${caption}</p>
                </div>
              </body>
            </html>`;
  res.send(html);
});
app.listen(process.env.PORT || 8090, () => {
console.log('App listening on port 8090');
});
```

In the above codes, port for local host will be enabled as 8090. Lets visualize how it works.

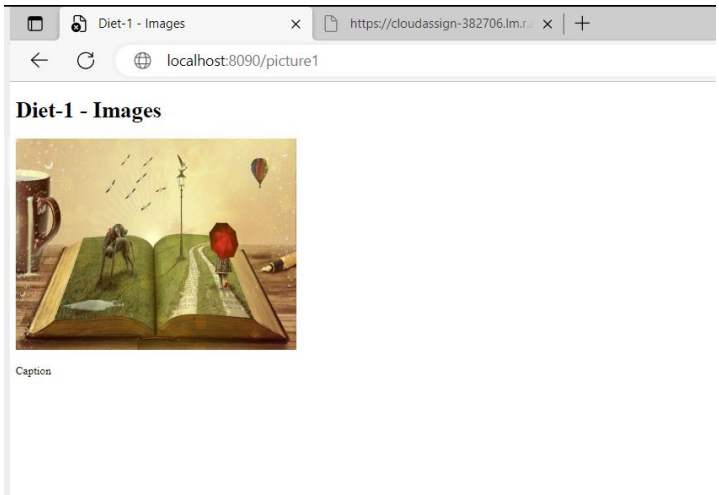
- Started app on **“localhost:8090”** by following command

Command:

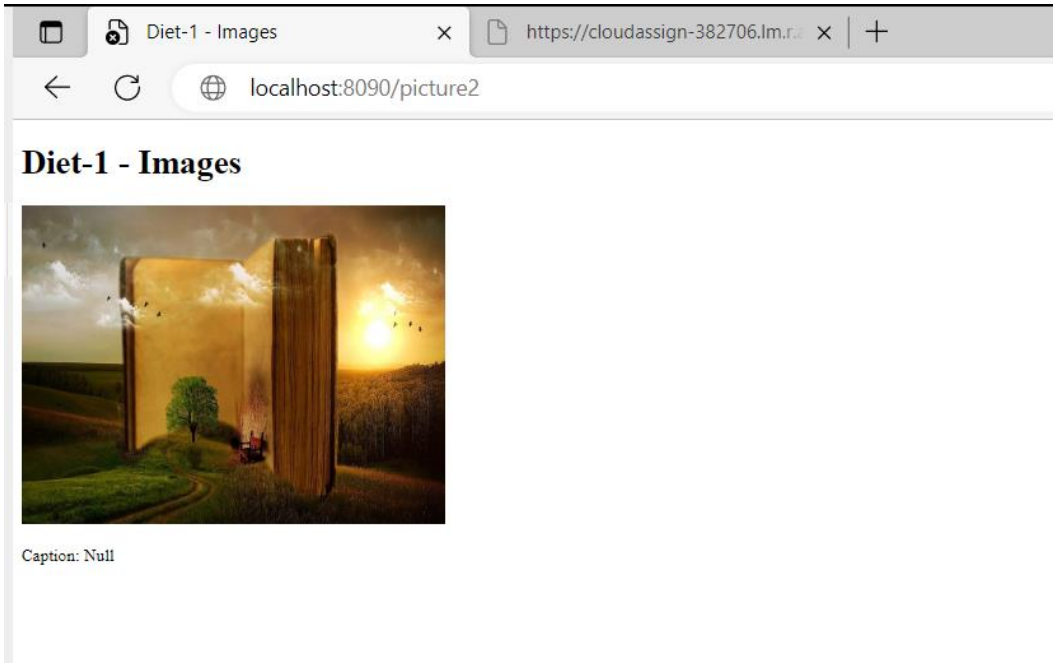
npm start



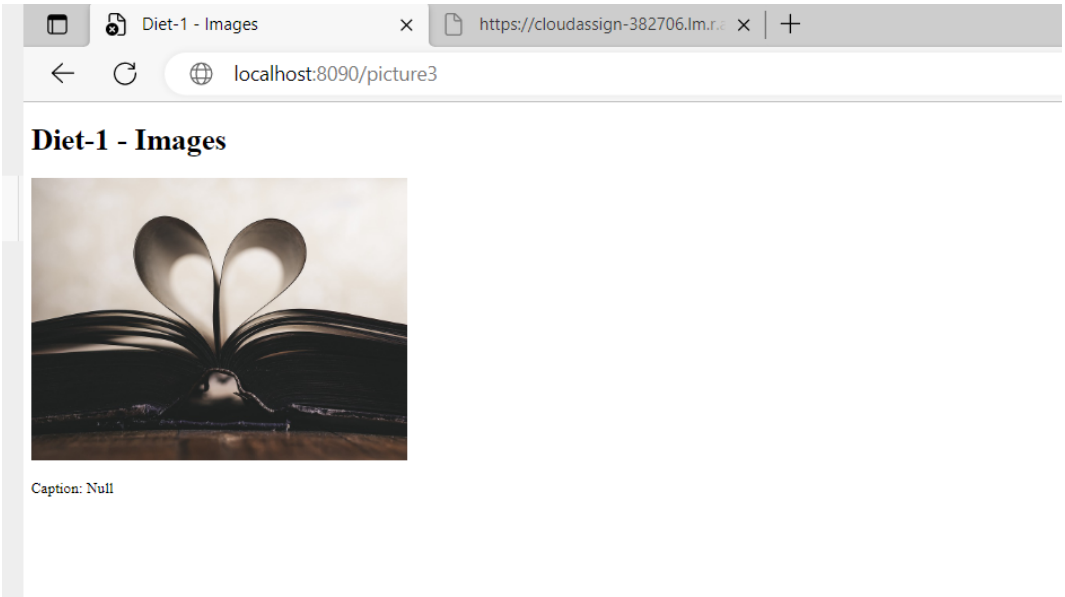
- Now Access the first Image



- Second Image



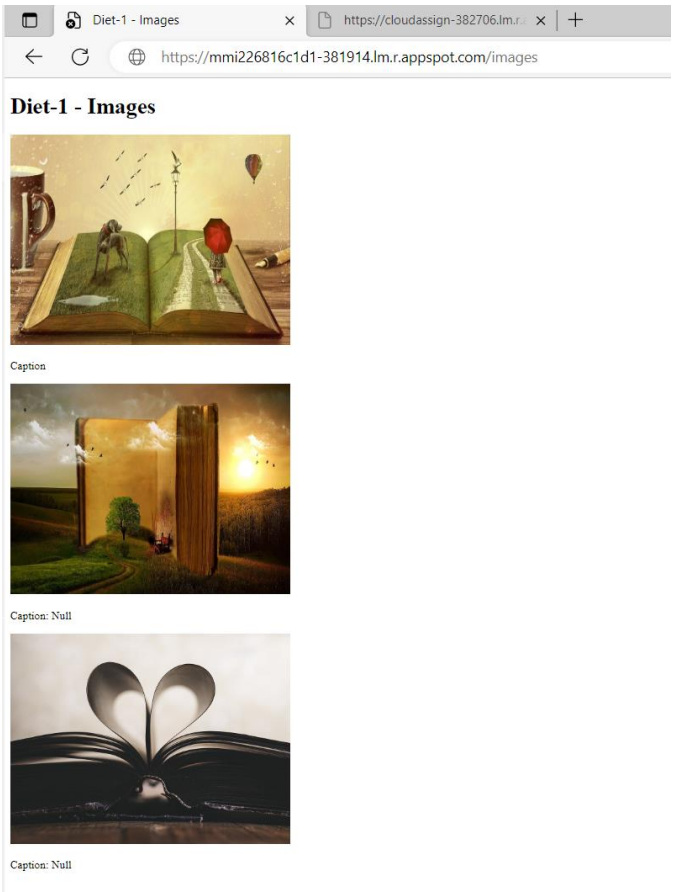
- Third Image



- AppEngine Deployment through Google Cloud

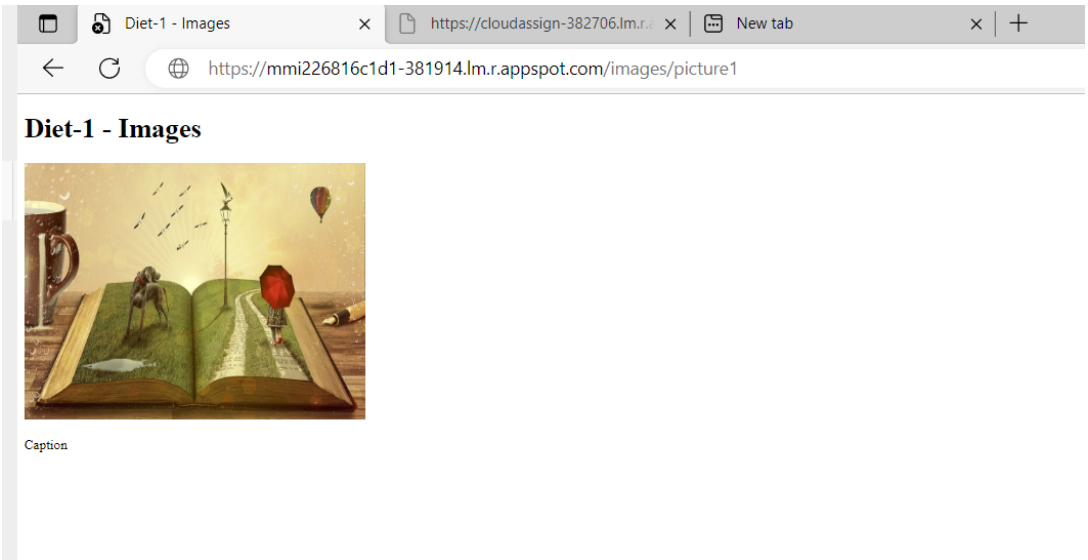
Command:
gcloud app deploy

- Accessing All Images Remotely: **cloudassign-382706.lm.r.appspot.com/images**



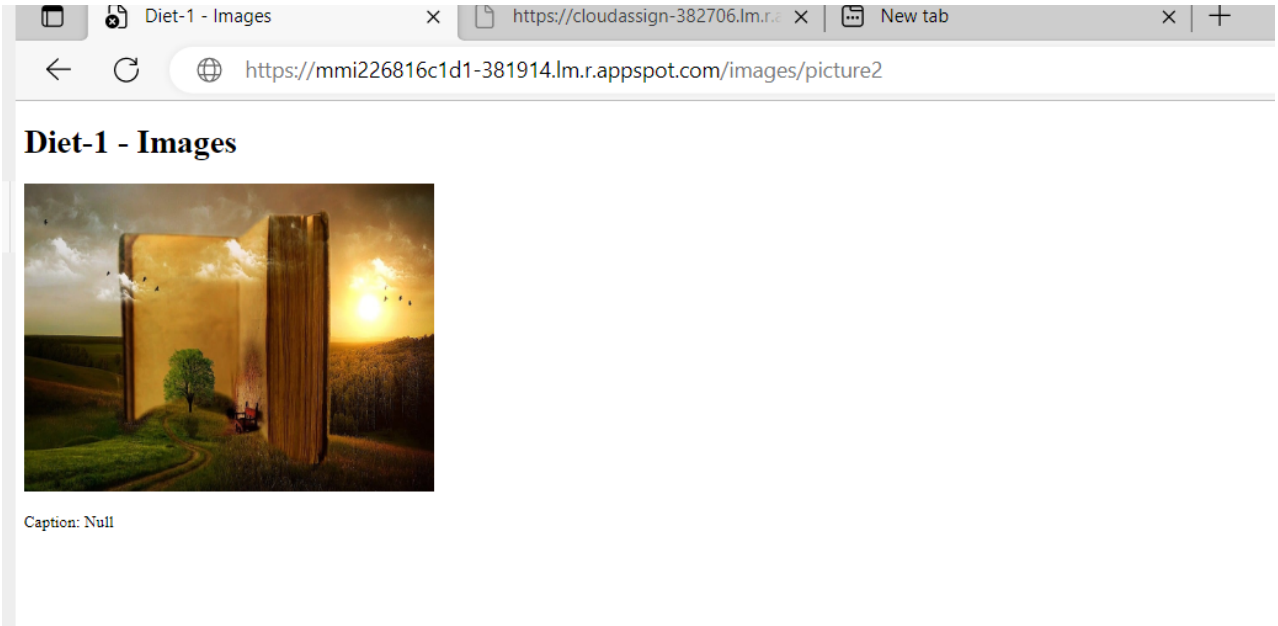
- Accessed First Image

Link: **cloudassign-382706.lm.r.appspot.com/images/picture1**



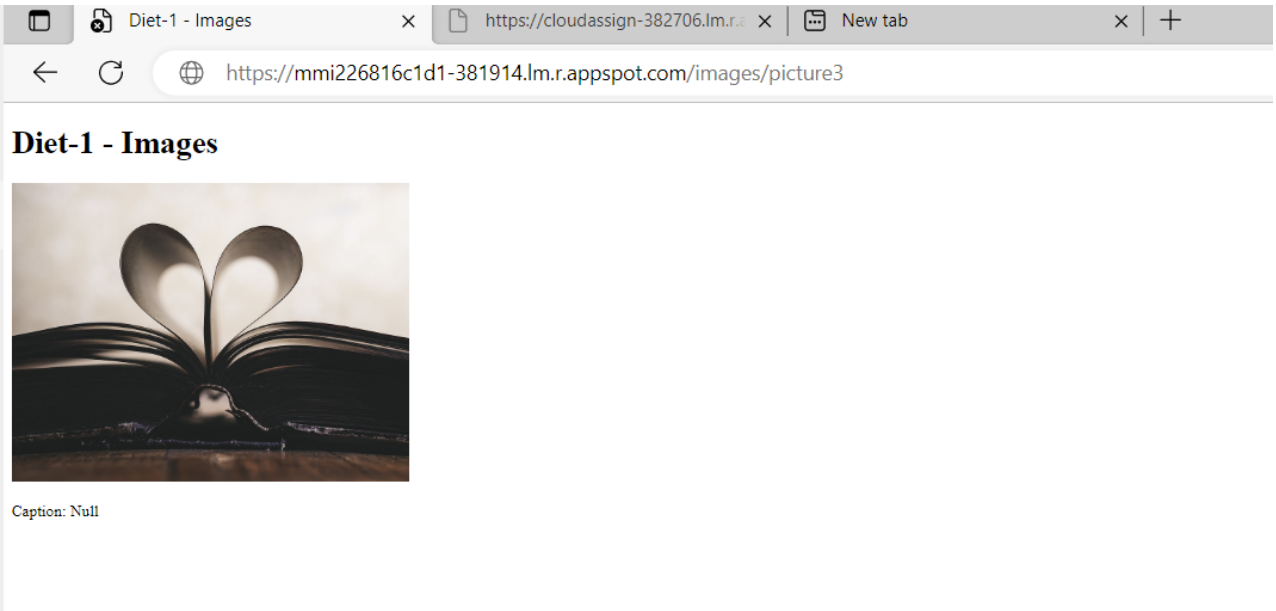
- Accessed Second Image Image

Link: **cloudassign-382706.lm.r.appspot.com/images/picture2**



- Accessed third Image

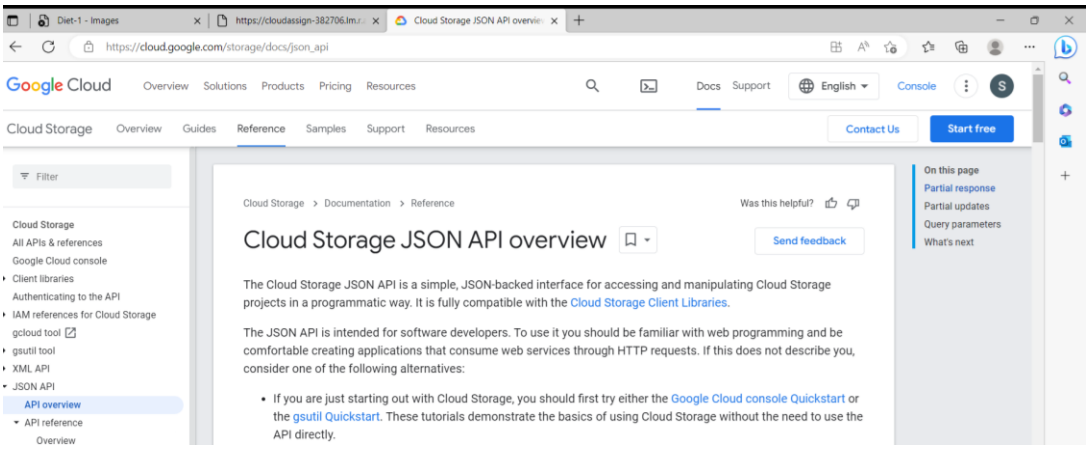
Link: **cloudassign-382706.lm.r.appspot.com/images/picture3**



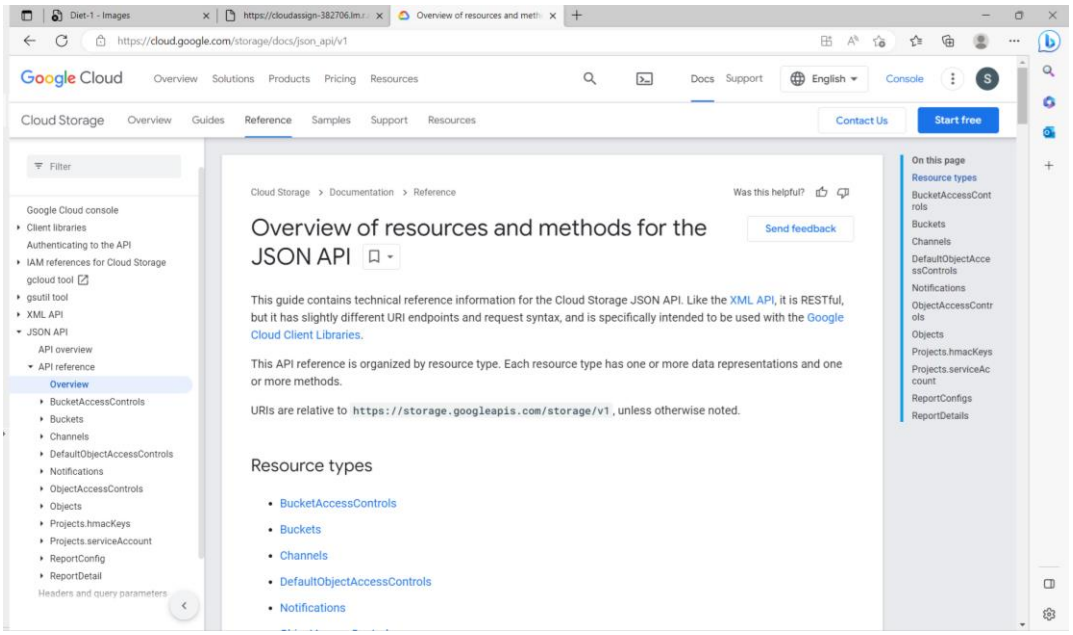
Task 3:

Part a: Explore Google API

Google Cloud API is a collection of Application Programming Interfaces (APIs) offered by Google Cloud Platform, which allows developers to access various Google Cloud services and resources programmatically. It provides a unified platform for managing and integrating various cloud services such as computing, storage, networking, and machine learning. Google Cloud API supports several programming languages and provides extensive documentation, code samples, and client libraries to facilitate easy integration. It also offers features such as authentication, authorization, monitoring, and logging to ensure secure and reliable communication between applications and Google Cloud services.



In accordance with our scenario, the "Cloud Storage API" is the most appropriate option as it offers support for both "Buckets" and "Objects". Upon further investigation of the "Objects" resource, we discovered a "get" method that can be utilized to obtain metadata pertaining to objects.



- Obtained the metadata for First image by this

storage.googleapis.com/storage/v1/b/diet1-bucket/o/picture2.jpg

Diet-1 - Images | storage.googleapis.com/storage/ | Overview of resources and meth

Not secure | storage.googleapis.com/storage/v1/b/diet1-bucket/o/picture2.jpg


```
{
"kind": "stage#object",
"id": "diet1-bucket/picture.jpg/1755191135172464",
"selflink":"https://www.googleapic.come/storage/v1/b/diet-bucket/o/picture2.jpg",
"medialink": "https://storage.googleapis.com/download/storage/v1/b/diet1/bucket/o/picture2.jpg?
generation=1755191135172464&alt=media",
"name": "picture2.jpg",
"bucket": "diet1-bucket",
"generation": "1755191135172464",
"metageneration" = "3",
"contentType": "image/jpg",
"storageClass": "STANDARD",
"size": "20035",
"mdHash": "w+w1ReD3Lf6LTQXTcvcrLw+=",
"contentLanguage": "en",
"crc32c": "dUygcA==",
"etag": "COXwkq+1sjdhsjX2EAM=",
"timeCreated": "2023-04-02T19:12:21.352Z",
"updated": "2023-04-02T21:14:21.832Z",
"timeStorageClassUpdated": "2023-04-02T19:12:21.352Z"
}.
```


- Metadata for Second Image

http://storage.googleapis.com/storage/v1/b/diet1-bucket/o/picture3.jpg

Diet-1 - Images | storage.googleapis.com/storage/ | Overview of resources and meth

storage.googleapis.com/storage/v1/b/diet1-bucket/o/picture3.jpg


```
{
"kind": "stage#object",
"id": "diet1-bucket/picture.jpg/1755191254865351",
"selflink":"https://www.googleapic.come/storage/v1/b/diet-bucket/o/picture3.jpg",
"medialink": "https://storage.googleapis.com/download/storage/v1/b/diet1/bucket/o/picture2.jpg?
generation=1755191254865351&alt=media",
"name": "picture3.jpg",
"bucket": "diet1-bucket",
"generation": "1755191254865351",
"metageneration" = "3",
"contentType": "image/jpg",
"storageClass": "STANDARD",
"size": "42332",
"mdHash": "ADFG/BNMKJ9adfFbOpRgh==",
"contentLanguage": "en",
"crc32c": "9G6a2B==",
"etag": "BVsMkL6hgfY8ASDF=",
"timeCreated": "2023-04-02T19:12:21.525Z",
"updated": "2023-04-02T21:14:21.900Z",
"timeStorageClassUpdated": "2023-04-02T19:12:21.525Z"
}.
```


- Metadata for third image

http://storage.googleapis.com/storage/v1/b/diet1-bucket/o/picture4.jpg

Diet-1 - Images | storage.googleapis.com/storage/ | Overview of resources and meth

storage.googleapis.com/storage/v1/b/diet1-bucket/o/picture4.jpg


```
{
"kind": "stage#object",
"id": "diet1-bucket/picture.jpg/1755191456598782",
"selflink":"https://www.googleapic.come/storage/v1/b/diet-bucket/o/picture3.jpg",
"medialink": "https://storage.googleapis.com/download/storage/v1/b/diet1/bucket/o/picture2.jpg?
generation=1755191456598782&alt=media",
"name": "picture4.jpg",
"bucket": "diet1-bucket",
"generation": "1755191456598782",
"metageneration" = "3",
"contentType": "image/jpg",
"storageClass": "STANDARD",
"size": "17365",
"mdHash": "QWE6Hah0r4c98HdW+aWhJU==",
"contentLanguage": "en",
"crc32c": "asZXCv==",
"etag": "KJywVGihA5ERTU=",
"timeCreated": "2023-04-02T21:14:22.112Z",
"updated": "2023-04-02T22:25:12.650Z",
"timeStorageClassUpdated": "2023-04-02T21:14:22.112Z"
}.
```


Part B: AppEngine development

Preferred Language: node Js

Command: npm init

Code:

```
{
    "name": "diet1-app3", "version":"3","description": "third app for third
taks",
    "main": "server.js",


    "scripts": {"test": "echo \"Error\" && exit"},
    "author": "shah hussain", "license": "ISC"
}
```

File Edit Selection View Go Run Terminal Help • package.json - Visual Studio Code

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

AppEngine.js | package.json | exp.jason | appEngine.yaml | NLP Analysis of Project Gutenberg Part 1.ipynb

```
1  {
2      "name": "diet1-app3", "version":"3","description": "third app for third taks",
3      "main": "server.js",
4
5
       Debug
6      "scripts": {"test": "echo \"Error\" && exit"},
7      "author": "shah hussain", "license": "ISC"
8  }
```

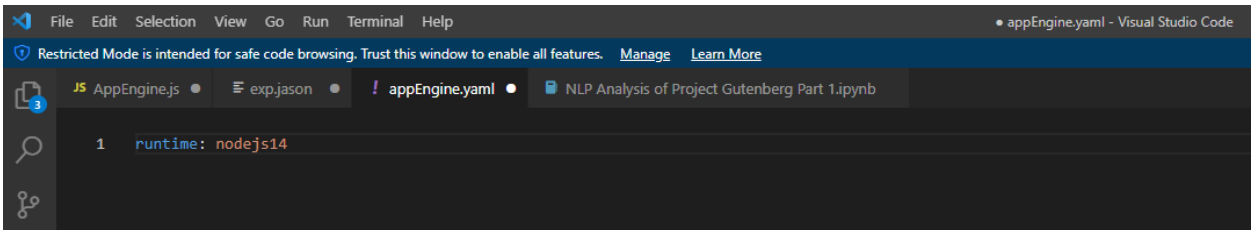
1. I have utilized the following command to install the npm package "express" and create a server within our newly created app:

```
npm install express
```

2. To request API endpoints, I installed the npm package "axios" using the following command:

```
npm install axios
```

3. I have created the "app.yaml" file with the following content.



- **Code for server.js**

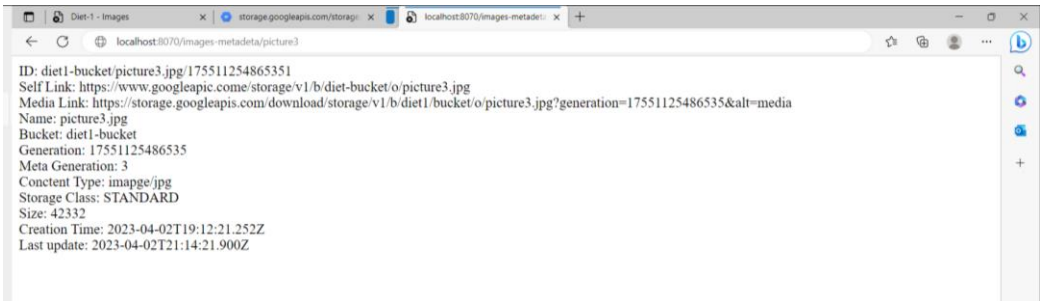
```
const express = require('express');

const axios = require('axios');

const app = express();

app.get('/images', (req, res) => {

  res.sendFile(__dirname + '/views/images.html');});

app.get('/images-metadata/:id', async (req, res) => {

  const imageId = parseInt(req.params.id);

  const url = `https://storage.googleapis.com/storage/v1/b/diet1-bucket/o/image${imageId}.png`;

  const response = await axios.get(url);

  if (response?.data) {    const metadata = response.data;

    const html = `<html><head><title>Image ${imageId}</title></head><body>

      ${Object.entries(metadata).map(([key, value]) => `<div><span>${key}</span>:
<span>${value}</span></div>`).join('')}

    </body></html>`;

    res.send(html);} else { const html = `<html><head><title>Image ${imageId}</title></head><body>

      <h1>No Image</h1> </body></html>`; res.send(html);  }});

const port = process.env.PORT || 8070;app.listen(port, () => {  console.log(`App listening on port
${port}`);});
```

- **App Testing: localhost:8070**

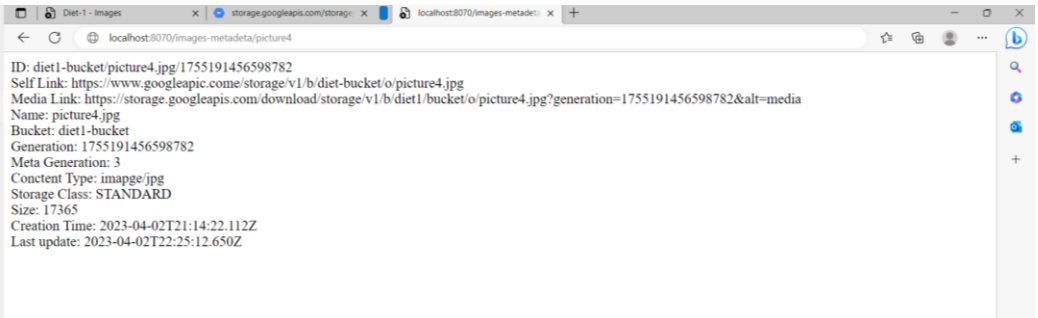
Command: npm start

- **Metadata for First image: http://localhost:8070/images-metadeta/picture1**

- **Metadata for Second Image: http://localhost:8070/images-metadeta/picture3**

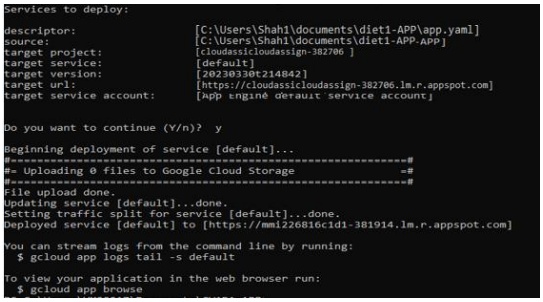


- **Metadata for third Image : http://localhost:8070/images-metadeta/picture4**



- **Deploying App through Google Cloud:**

Command: gcloud app deploy



```
Services to deploy:

descriptor:                  [C:\Users\Shah1\documents\diet1-APP\app.yaml]
source:                      [C:\Users\Shah1\documents\diet1-APP-APP]
target project:              [cloudassicloudassign-382706 ]
target service:              [default]
target version:              [20230330t214842]
target url:                  [https://cloudassicloudassign-382706.lm.r.appspot.com]
target service account:      [App Engine default service account]


Do you want to continue (Y/n)?  y

Beginning deployment of service [default]...
#=================================================================#
#= Uploading 0 files to Google Cloud Storage                     =#
#=================================================================#
File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://mmi226816c1d1-381914.lm.r.appspot.com]

You can stream logs from the command line by running:
  $ gcloud app logs tail -s default

To view your application in the web browser run:
  $ gcloud app browse
```

- **Metadata for first image:** cloudassign-382706.lm.r.appspot.com/images-metadata/picture2

Diet-1 - Images | storage.googleapis.com/storage/ | Picture2

Not secure | cloudassign-382706.lm.r.appspot.com/images-metadata/picture2

diet1-bucket/picture2.jpg/1755191135172464
Self Link: https://www.googleapic.come/storage/v1/b/diet-bucket/o/picture2.jpg
Media Link: https://storage.googleapis.com/download/storage/v1/b/diet1/bucket/o/picture2.jpg?generation=1755191135172464&alt=media
Name: picture2.jpg
Bucket: diet1-bucket
Generation: 1755191135172464
Meta Generation: 3
Conctent Type: imapge/jpg
Storage Class: STANDARD
Size: 20035
Creation Time: 2023-04-02T19:12:21.352Z
Last update: 2023-04-02T21:14:21.832Z

- **Metadata for second image:** cloudassign-382706.lm.r.appspot.com/images-metadata/picture3

Diet-1 - Images | storage.googleapis.com/storage/ | Picture3

Not secure | cloudassign-382706.lm.r.appspot.com/images-metadata/picture2

diet1-bucket/picture2.jpg/1755191456598782
Self Link: https://www.googleapic.come/storage/v1/b/diet-bucket/o/picture2.jpg
Media Link: https://storage.googleapis.com/download/storage/v1/b/diet1/bucket/o/picture2.jpg?generation=1755191456598782&alt=media
Name: picture3.jpg
Bucket: diet1-bucket
Generation: 1755191456598782
Meta Generation: 3
Conctent Type: imapge/jpg
Storage Class: STANDARD
Size: 42332
Creation Time: 2023-04-02T19:12:21.252Z
Last update: 2023-04-02T21:14:21.900Z

- **Metadata for third image:** cloudassign-382706.lm.r.appspot.com/images-metadata/picture4

Diet-1 - Images | storage.googleapis.com/storage/ | Picture4

cloudassign-382706.lm.r.appspot.com/images-metadata/picture4

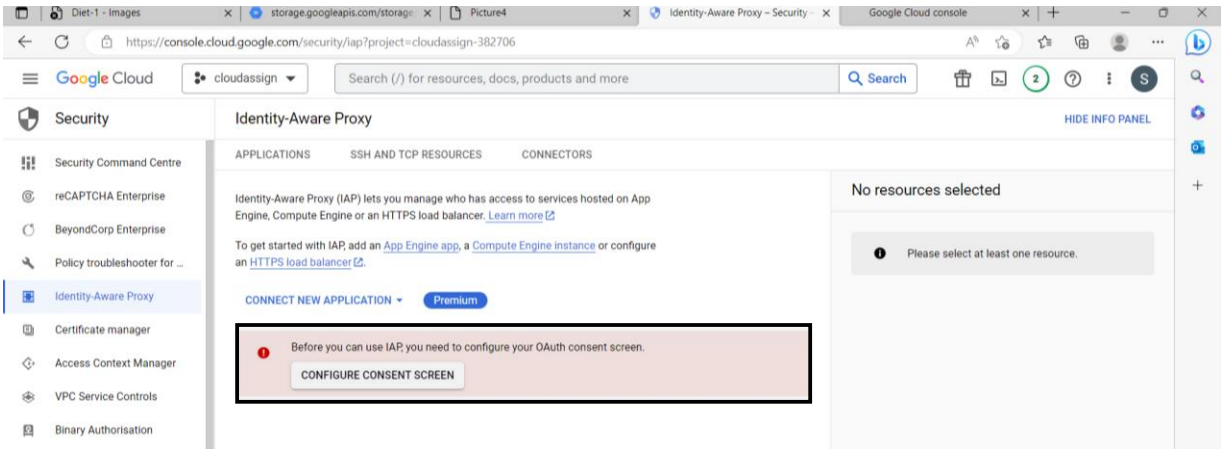
diet1-bucket/picture2.jpg/1755191456598782
Self Link: https://www.googleapic.come/storage/v1/b/diet-bucket/o/picture2.jpg
Media Link: https://storage.googleapis.com/download/storage/v1/b/diet1/bucket/o/picture4.jpg?generation=1755191456598782&alt=media
Name: picture4.jpg
Bucket: diet1-bucket
Generation: 1755191456598782
Meta Generation: 3
Conctent Type: imapge/jpg
Storage Class: STANDARD
Size: 17365
Creation Time: 2023-04-02T21:14:22.112Z
Last update: 2023-04-02T22:25:12.650Z

Part C: Securing App Engine through IAP:

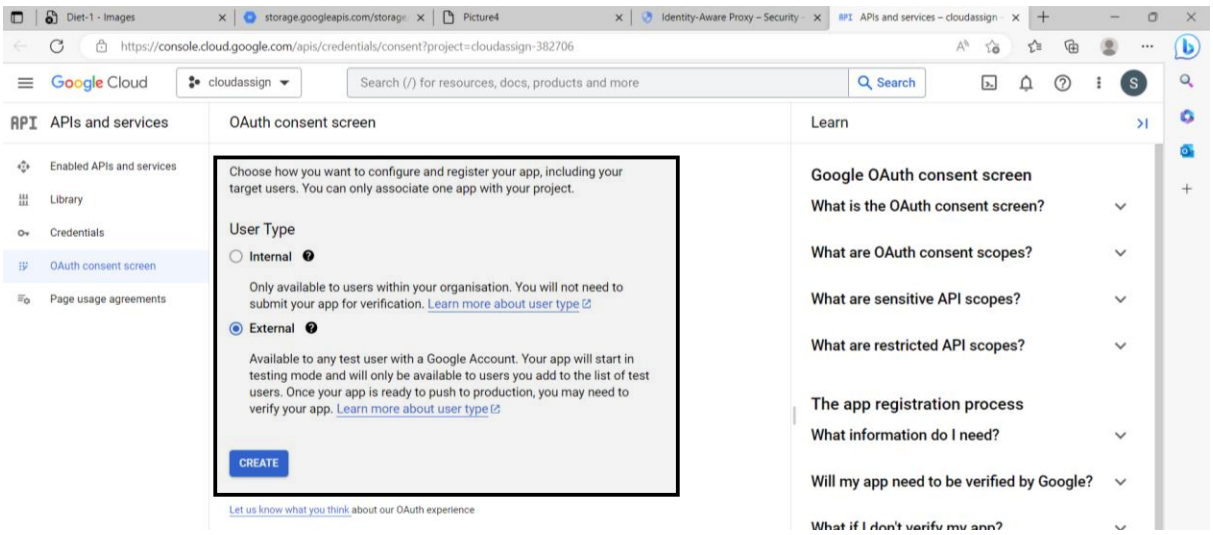
Securing App Engine with Identity-Aware Proxy (IAP) provides several benefits, such as enhancing the overall security of the application by adding an extra layer of authentication and authorization. IAP enables granular access control to specific resources and provides centralized management of access policies, making it easier to manage and enforce security policies across an organization. Additionally, IAP allows users to access applications from any device, location, or network while ensuring secure access to sensitive data. By using IAP to secure App Engine, organizations can reduce the risk of unauthorized access, data breaches, and other security threats.

Step 1: Go to https://console.google.com/security

Then opened IAP tag for OAuth consent scree.



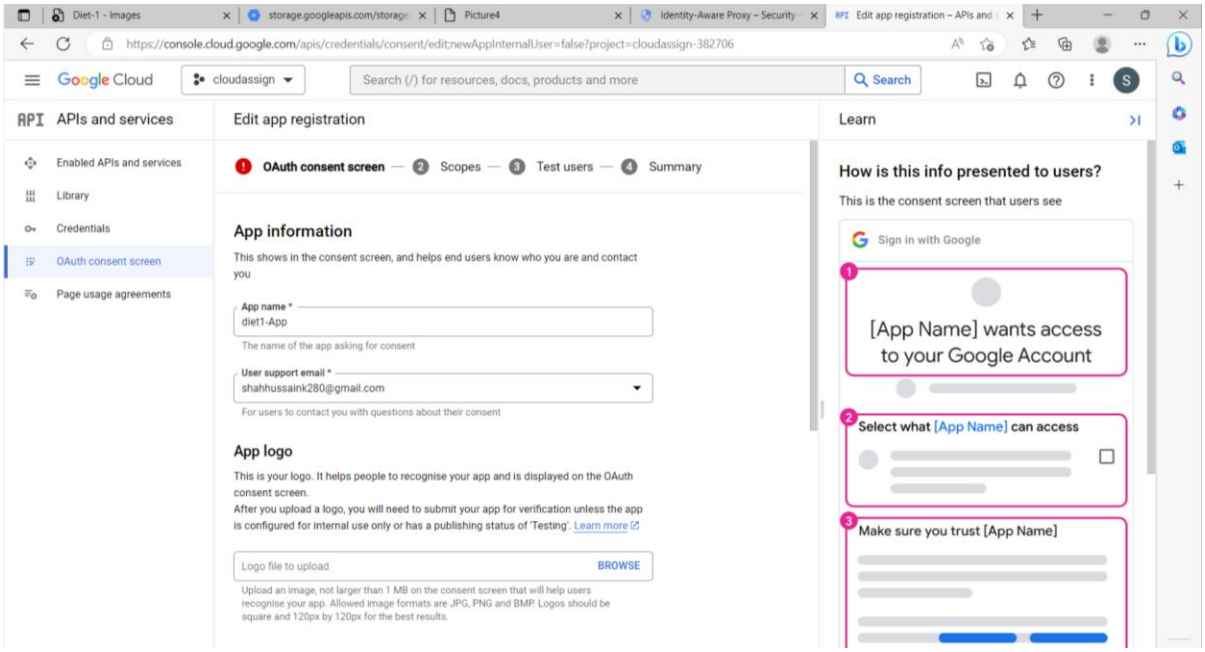
Here, selected the External option for the users of any kind with a google account.



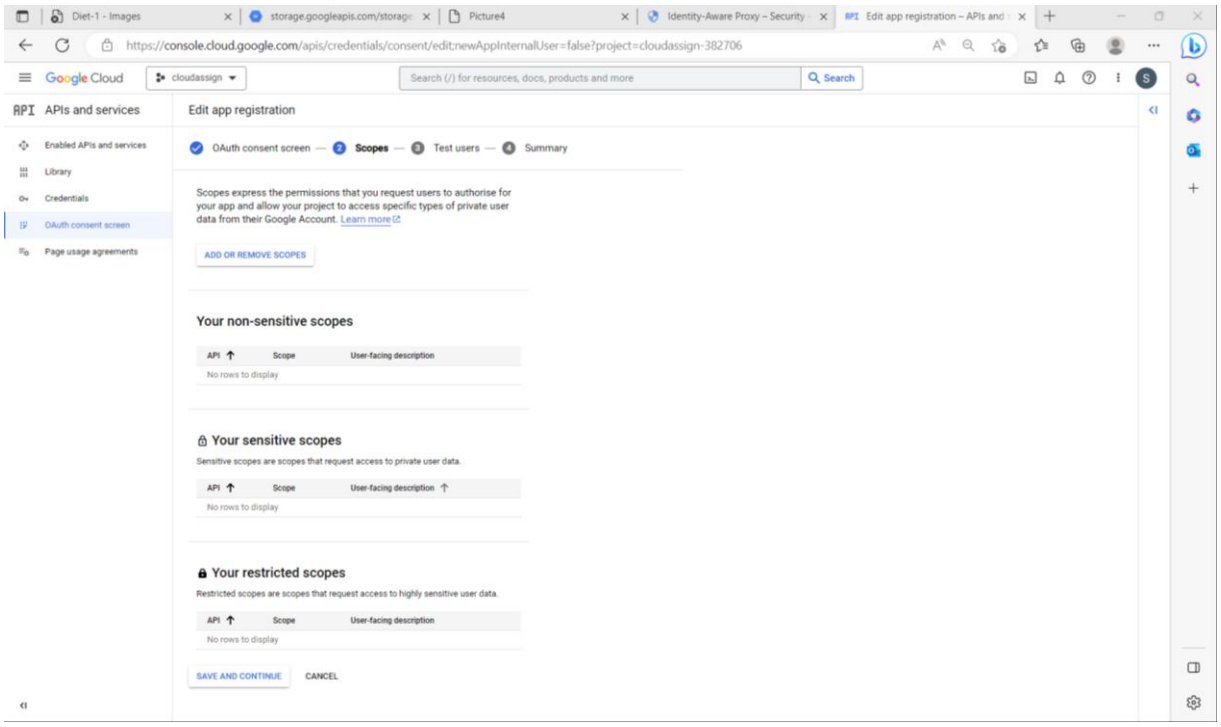
In Next Step, I have added App information

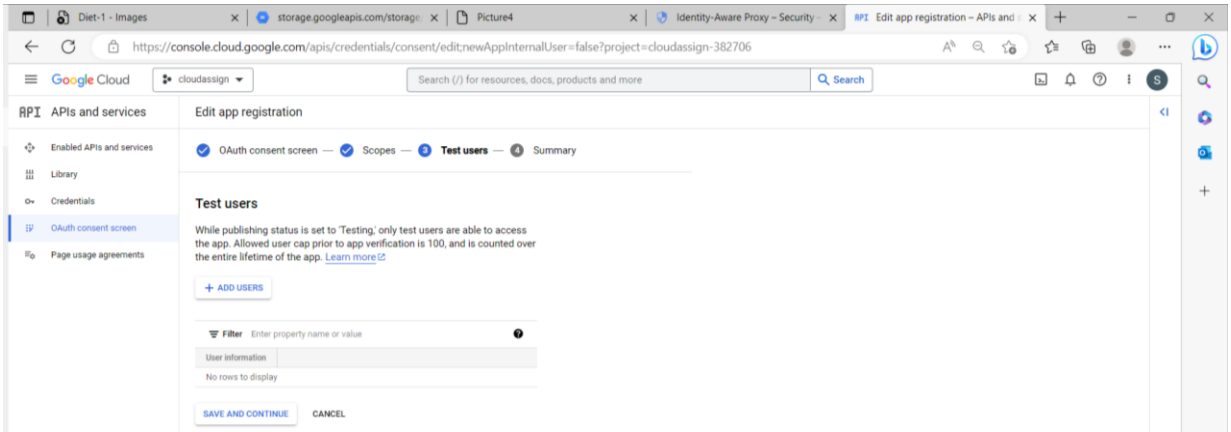
App Name: diet1-App

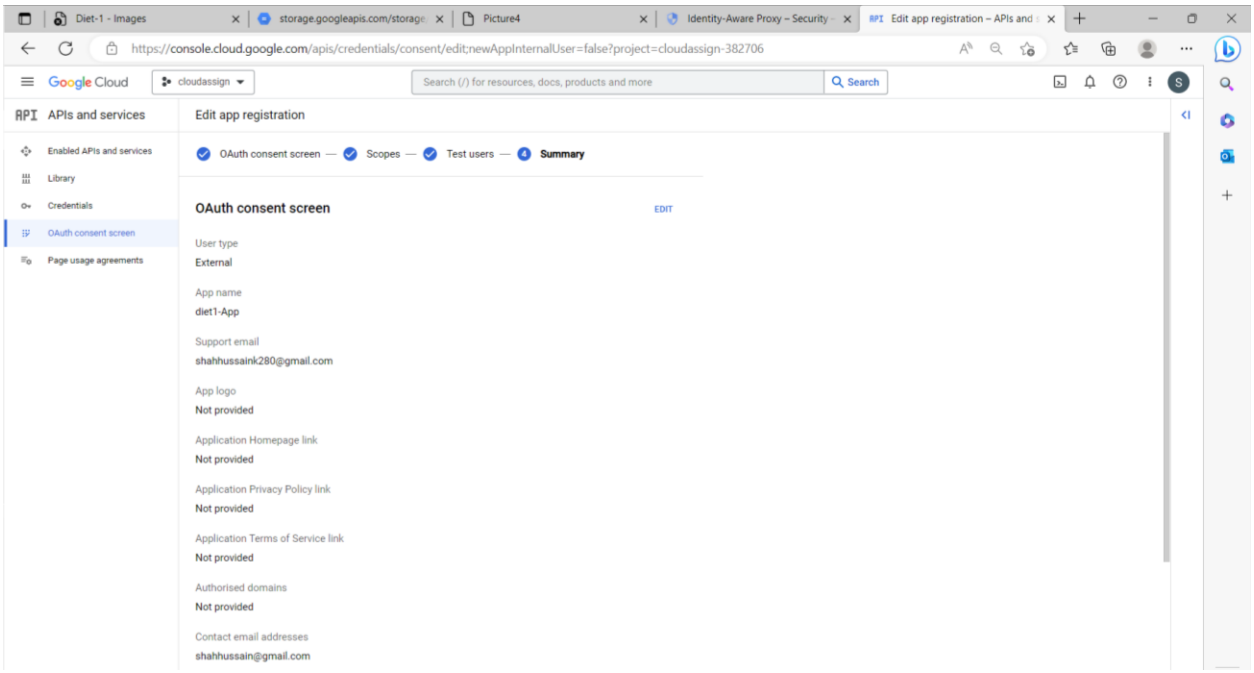
User support name: shahhussain280@gmail.com



Next Steps have been highlighted in terms of screen shorts







Summary of Task 3-Part C:

- To secure App Engine app using Google Identity-Aware Proxy (IAP), begin by creating a new project or selecting an existing one in the Google Cloud Console.
- Deploy App Engine app to the project using either the gcloud command-line tool or the Google Cloud Console.
- Enable IAP for the App Engine app by navigating to the "Identity-Aware Proxy" page in the Cloud Console and selecting App Engine app. Follow the prompts to enable IAP.
- To grant access to the App Engine app, create an Identity-Aware Proxy access policy and add user as an authorized user.
- Test the App Engine app by navigating to its URL in a browser. I should be redirected to the Google sign-in page, where you'll need to enter your Google account credentials.
- Once you've entered your credentials, you should be able to access the App Engine app.