Greeshmika Korrapati - U00932594

Cloud Computing Project - 3

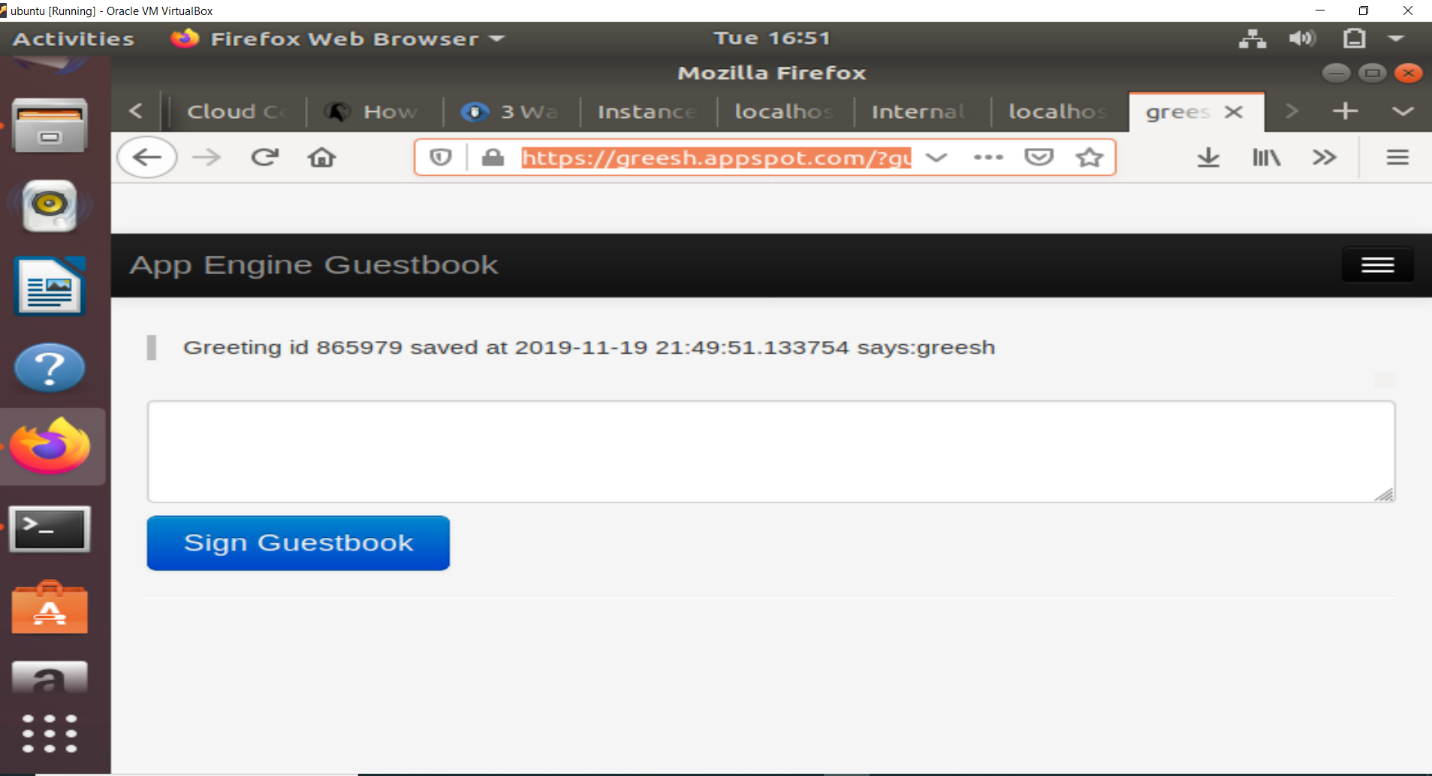
1.1

In the downloaded Guestbook code there are two index files index.yaml and index.yaml 1.I have deleted deleted index.yaml 1 before deleting I have copied content in that and pasted in index.yaml. Also, as per requirement revised application by changing code to Greeting id {{ greeting.gid }} saved at {{ greeting.date }} says: {{ greeting.content }} . By this it displays greeting id, greeting date and greeting content.

My URL link for GAE application : <https://greesh.appspot.com/?guestbook_name=default_guestbook>.

After deploying application by using gcloud app deploy index.yaml , I was getting index error so used gcloud app deploy index.yaml. This has fixed error and app is running successfully.

Successfully deployed app and below is the screenshot.



1.2

Created greetings table in DynamoDB and inserted new records into it with help of add\_item function, get\_table function which returns table is used in all other functions. Delete\_item is used to delete records from table. I have successfully created table, added new items, read records, delete records.

CODE:

from \_\_future\_\_ import print\_function

import boto3

from boto3.dynamodb.conditions import Key

from boto3 import resource

def create\_table(table\_name):

dynamodb\_resource = resource('dynamodb',region\_name='us-east-2', aws\_access\_key\_id ='AKIAIPCD6YLRSLDTRBRQ', aws\_secret\_access\_key='YuY+58G682xsAtPURpQd/YJE/eh6ElX7ynnnBpij')

# to do

# check the sample code https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.01.html

# create the greetings table with attributes (gid, date, content).

# return the table object

mytable = dynamodb\_resource.create\_table(

TableName=table\_name,

KeySchema=[

{

'AttributeName': 'gid',

'KeyType': 'HASH' #Partition key

},

],

AttributeDefinitions=[

{

'AttributeName': 'gid',

'AttributeType': 'S'

},

],

ProvisionedThroughput={

'ReadCapacityUnits': 10,

'WriteCapacityUnits': 10

}

)

mytable.meta.client.get\_waiter('table\_exists').wait(TableName='greetings')

print(mytable.item\_count)

print("Table status:", mytable.table\_status)

def get\_table(table\_name):

"return the table object, when the table is already created"

dynamodb\_resource = resource('dynamodb',region\_name='us-east-2', aws\_access\_key\_id ='AKIAIPCD6YLRSLDTRBRQ', aws\_secret\_access\_key='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'))

table = None

try:

table = dynamodb\_resource.Table(table\_name)

except:

print ("cannot get the table", table\_name)

finally:

return table

def read\_table\_item(table, pk\_name, pk\_value):

"""

table is the object returned by get\_table

Return item read by primary key.

"""

tabledata = get\_table(table)

response = tabledata.get\_item(Key={pk\_name: pk\_value})

return response

def add\_item(table, col\_dict):

"""

Add one item (row) to table. col\_dict is a dictionary {col\_name: value}.

"""

tabledata = get\_table(table)

response = tabledata.put\_item(Item=col\_dict)

return response

def delete\_item(table, pk\_name, pk\_value):

"""

Delete an item (row) in table from its primary key.

"""

tabledata = get\_table(table)

response = tabledata.delete\_item(Key={pk\_name: pk\_value})

return response

if \_\_name\_\_ == '\_\_main\_\_':

create\_table("greetings")

#item1 = {'gid': '52222', 'date': '11/19/2019', 'content': 'first record'}

#item2 = {'gid': '53332', 'date': '11/20/2019', 'content': 'second record'}

#add\_item("greetings",item1)

#print("first record added")

#add\_item("greetings", item2)

#print("second record added")

#read\_table\_item("greetings",'gid','52222')

#print("Reading done successfully")

#delete\_item("greetings",'gid','52222')

#print("Deleted successfully")

2.1

Microservices code:

from flask import Flask

from werkzeug.exceptions import NotFound

from flask import make\_response

import json

import dynamo # the code you finished for Part I

app = Flask(\_\_name\_\_)

# code here to open the DynamoDB table. If the table is not there, create it

# to do

def root\_dir():

""" Returns root director for this project """

return os.path.dirname(os.path.realpath(\_\_file\_\_ + '/..'))

def nice\_json(arg):

response = make\_response(json.dumps(arg, sort\_keys = True, indent=4))

response.headers['Content-type'] = "application/json"

return response

@app.route("/", methods=['GET'])

def hello():

return nice\_json({

"uri": "/",

"subresource\_uris": {

"greetings": "/greetings",

"add\_greeting": "/greetings/<id>/<date>/<content>",

}

})

@app.route("/greetings", methods=['GET'])

def greetings():

list=[]

table=dynamo.get\_table("greetings")

tablecontent=table.scan()

for content in tablecontent['Items']:

list.append(content)

return nice\_json(list)

@app.route("/addgreeting/<gid>/<date>/<content>", methods=['POST', 'PUT'])

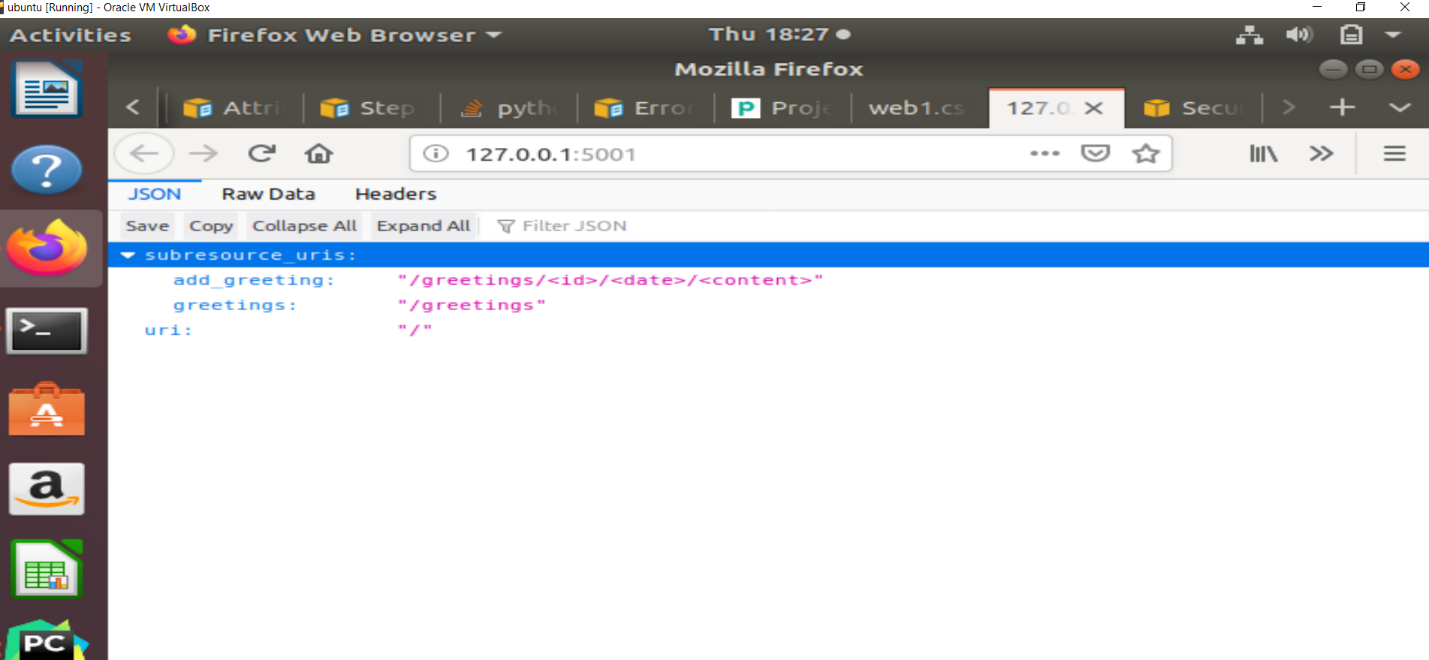
def add\_greeting(gid, date, content):

table\_dict={'gid':gid,'date':date,'content':content}

return nice\_json(dynamo.add\_item("greetings",table\_dict))

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0',port=5001, debug=True)



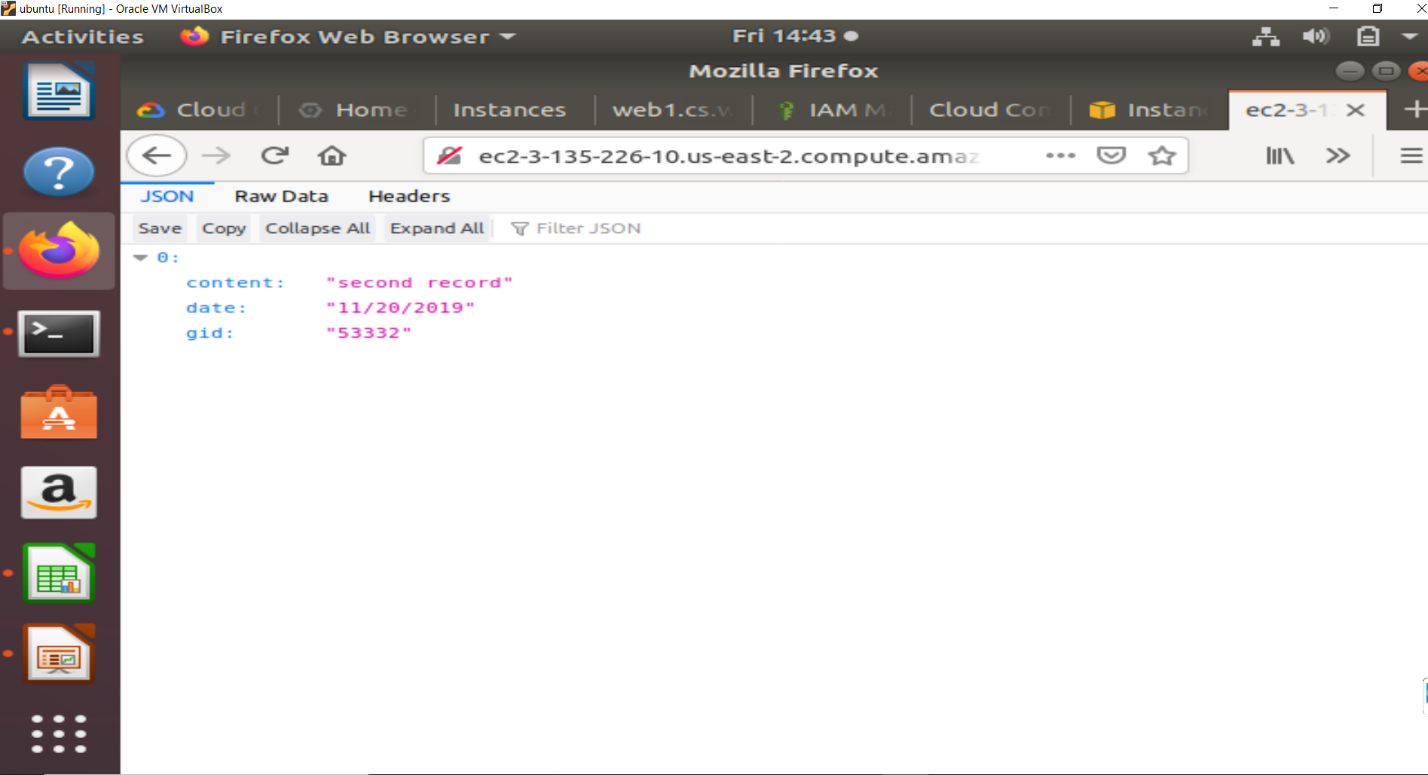
Deploying into Ec2 instance:

I have created instance in AWS and connected with help of ssh using my pem key.

**ssh -i "greeshknew.pem"** [**ubuntu@ec2-3-135-226-10.us-east-2.compute.amazonaws.com**](mailto:ubuntu@ec2-3-135-226-10.us-east-2.compute.amazonaws.com)**.**

Created folder flashapp and copied both microservies.py file and dynamo.py file which I have used in 1.2 in this flashapp folder.Then I have installed flash,boto3 .Then I ran my code as python3.microservices.py

Output can be seen in browser : **ec2-3-135-226-10.us-east-2.compute.amazonaws.com:5001/greetings**



For testing I have used test.py

import requests

from requests.exceptions import HTTPError

for url in ['http://ec2-3-135-226-10.us-east-2.compute.amazonaws.com:5001/greetings']:

try:

response = requests.get(url)

print(response)

response.raise\_for\_status()

except HTTPError as http\_err:

print('Http error occurred:{http\_err}')

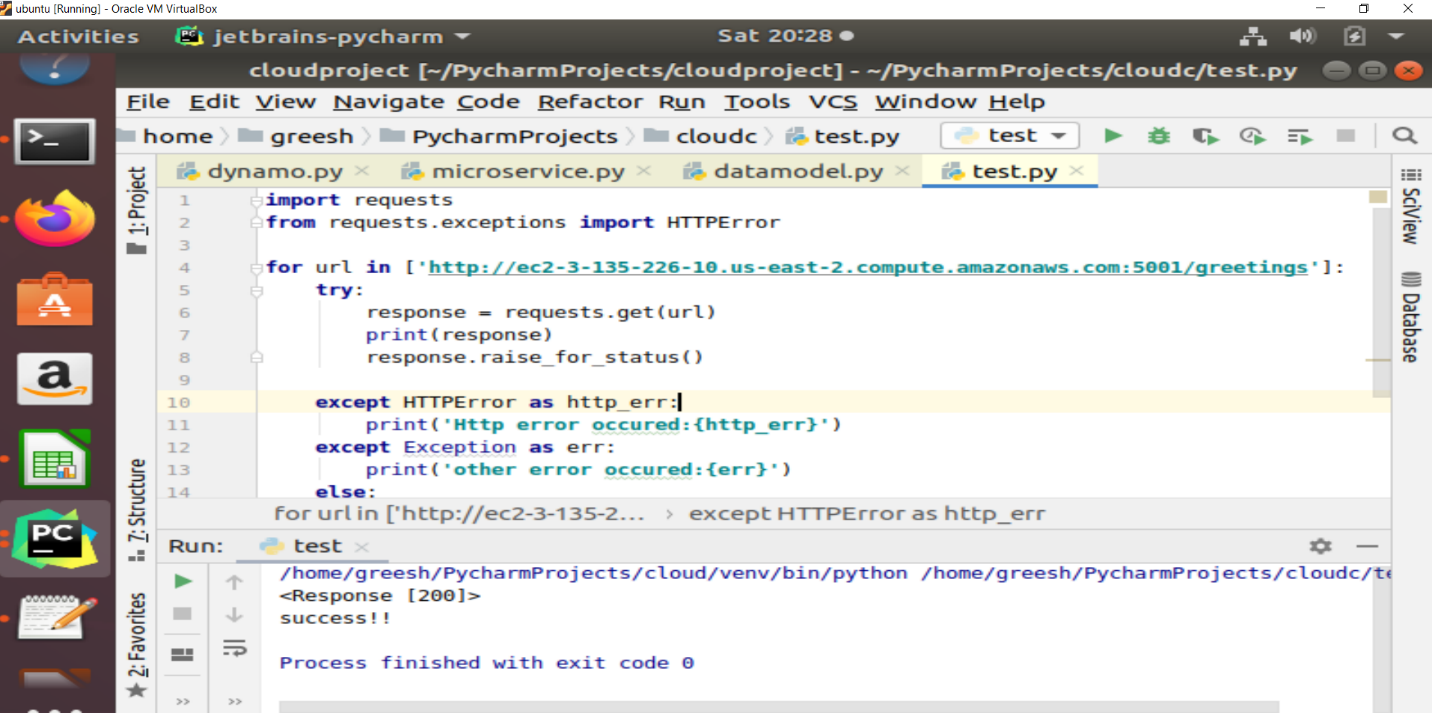
except Exception as err:

print('other error occurred:{err}')

else:

print('success!!')

I have tested microservices in two ways one is logging with SSH and running microservices and checking url for both get and post. Second way is that I wrote test.py where I have used python requests. This is a simple python file where I have given URl and shows status of it and prints success!!.If there are exceptions it prints exception messages respectively.



2.2

We must use datamodel.py for this and it is implemented in 2.3. In addition to this we have do few changes in guestbook.py as well. We have to import datamodel.py UnifiedGreeting class in guestbook.py. In MainPage class of guestbook.py we have get function it has to be modified such that it calls UnifiedGreeting getGreetings() function. In this function dynamoGreeting getGreetings() is called.guestbook.py has post method this has to be changed such that UnifiedGreeting addGreetings() function is called and in that function dynamoGreeting addGreetings() is called.In this way GAE url fetch API can access microservice to add/retrieve greetings.

2.3

CODE:

datamodel:

import abc

from greeting import Greeting

import json

import webapp2

from google.appengine.ext import ndb

from google.appengine.api import urlfetch

# the base class

class GreetingModel:

\_\_metaclass\_\_ = abc.ABCMeta

@abc.abstractmethod

def getGreetings(self):

pass

@abc.abstractmethod

def addGreeting(self, gid, date, content):

pass

class GAEGreeting(GreetingModel):

def \_\_init\_\_(self, guestbook\_name):

# constructor, initialize anything you need

# to do

self.guestbook\_name = guestbook\_name

pass

def getGreetings(self):

# to do

greetings\_query = Greeting.query(ancestor=self.guestbook\_key()).order(-Greeting.date)

greetings\_records= greetings\_query.fetch(10)

return greetings\_records

def guestbook\_key(self,):

return ndb.Key('Guestbook',self.guestbook\_name)

def addGreeting(self, gid, date, content):

# to do

newgreeting = Greeting(parent = self.guestbook\_key())

newgreeting.gid = gid

newgreeting.content = content

newgreeting.put()

return newgreeting.date

class DynamoGreeting(GreetingModel):

def \_\_init\_\_(self, guestbook\_name):

# to do

pass

def getGreetings(self):

# to do

try:

mylink ="http://ec2-3-135-226-10.us-east-2.compute.amazonaws.com:5001/greetings"

greeting\_records = urlfetch.fetch (mylink, method = urlfetch.GET)

display= json.loads(greeting\_records.content)

print(display)

for eachrecord in display:

print(eachrecord)

except urlfetch.Error:

print("Sorry !!! Error in loading page")

return display

def addGreeting(self, gid, date, content):

newdate=newdate=date.replace(" ","0")

newlink = "http://ec2-3-135-226-10.us-east-2.compute.amazonaws.com:5001/addgreeting/"+str(gid)+"/"+str(newdate)+"/"+str(content)

required\_record=urlfetch.fetch(newlink,method=urlfetch.POST)

print(required\_record.content)

class UnifiedGreeting(GreetingModel):

def \_\_init\_\_(self, guestbook\_name):

# create both GAE and Dynamo Models

# the UnifiedGreeting model will be used by the GAE main program

# to do

pass

def getGreetings(self):

# pick one model to get all greetings

# to do

dynamo\_record =DynamoGreeting('default\_guestbook')

greeting\_record = dynamo\_record.getGreetings()

return greeting\_record

def addGreeting(self, gid, date, content):

# append the new record to both models

# to do

gae\_data= GAEGreeting('default\_guestbook')

record\_new= gae\_data.addGreeting(gid,date,content)

dynamo\_data = DynamoGreeting('default\_guestbook')

new\_date = str(record\_new)

dynamo\_data.addGreeting(gid,new\_date,content)

guestbook:

import os

import urllib

#import boto3

import sys

sys.platform = 'linux3'

from google.appengine.api import users

from google.appengine.api import urlfetch

from google.appengine.ext import ndb

import jinja2

import webapp2

#from greeting import Greeting

import random

from datamodel import UnifiedGreeting

JINJA\_ENVIRONMENT = jinja2.Environment(

loader=jinja2.FileSystemLoader(os.path.dirname(\_\_file\_\_)),

extensions=['jinja2.ext.autoescape'],

autoescape=True)

# [END imports]

DEFAULT\_GUESTBOOK\_NAME = 'default\_guestbook'

# We set a parent key on the 'Greetings' to ensure that they are all

# in the same entity group. Queries across the single entity group

# will be consistent. However, the write rate should be limited to

# ~1/second.

def guestbook\_key(guestbook\_name=DEFAULT\_GUESTBOOK\_NAME):

"""Constructs a Datastore key for a Guestbook entity.

We use guestbook\_name as the key.

"""

return ndb.Key('Guestbook', guestbook\_name)

# [START main\_page]

class MainPage(webapp2.RequestHandler):

def get(self):

guestbook\_name = self.request.get('guestbook\_name',

DEFAULT\_GUESTBOOK\_NAME)

unified\_greetings = UnifiedGreeting(guestbook\_name)

greetings\_new = unified\_greetings.getGreetings()

template\_values = {

'greetings': greetings\_new,

'guestbook\_name': guestbook\_name

}

template = JINJA\_ENVIRONMENT.get\_template('index.html')

self.response.write(template.render(template\_values))

# [END main\_page]

# [START guestbook]

class Guestbook(webapp2.RequestHandler):

def post(self):

# We set the same parent key on the 'Greeting' to ensure each

# Greeting is in the same entity group. Queries across the

# single entity group will be consistent. However, the write

# rate to a single entity group should be limited to

# ~1/second.

guestbook\_name = self.request.get('guestbook\_name',

DEFAULT\_GUESTBOOK\_NAME)

gid\_required =random.randint(0,1000000)

date\_required = None

content\_required = self.request.get('content')

try:

unified\_greeting = UnifiedGreeting(guestbook\_name)

unified\_greeting.addGreeting(gid\_required,date\_required,content\_required)

except urlfetch.DownloadError:

self.response.write("Sorry!! Unable to load the page")

except urlfetch.InvalidURLError:

self.response.write("Please check your Url,it is not valid")

query\_params = {'guestbook\_name': guestbook\_name}

self.redirect('/?' + urllib.urlencode(query\_params))

# [END guestbook]

# [START app]

app = webapp2.WSGIApplication([

('/', MainPage),

('/sign', Guestbook),

], debug=True)

# [END app]

I have included datamodel.py in guestbook folder and imported datamodel UnifiedGreeting class in guestbook.py. In guestbook.py Mainpage class is modified such that it uses UnifiedGreeting class get method to access any one of DynamoGreeting getGreetings function or GAEGreeting getGreetings function. Here I have used DynamoGreeting getGreetings to fetch all greetings and display in html page.

POST method in guestbook.py is modified such that it helps to store new greetings in both DynamoDB and in Google App Engine.In this method UnifiedGreeting class , addGreeting is called ,this function calls both addGreeting functions of Dynamodb as well as Google App Engine and updates the new greetings successfully.