Greeshmika Korrapati - U00932594

Cloud Computing Project - 3

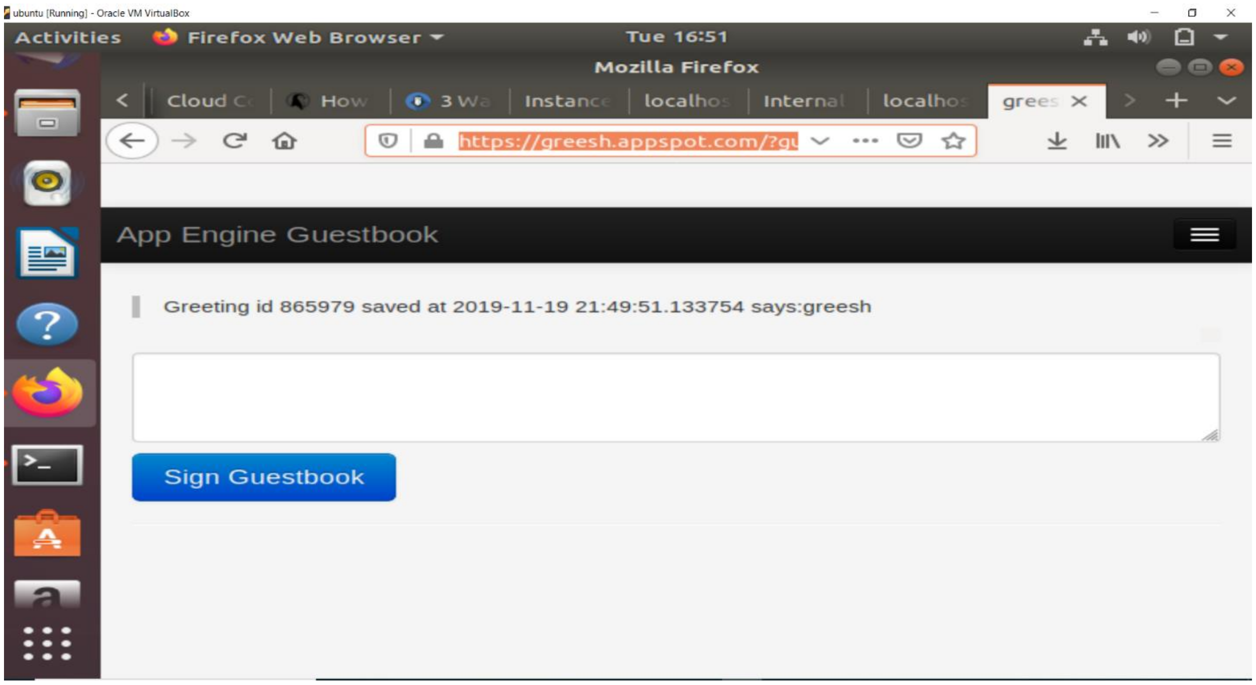
1.1

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.



Greeting id 865979 saved at 2019-11-19 21:49:51.133754 says:greesh

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')
    # to do
    # check the sample code
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Getti
ngStarted.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='greetin
gs')
    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')
    table = None
    try:
        table = dynamodb_resource.Table(table_name)
    except:
        print ("cannot get the table", table_name)
    finally:
        return table
```

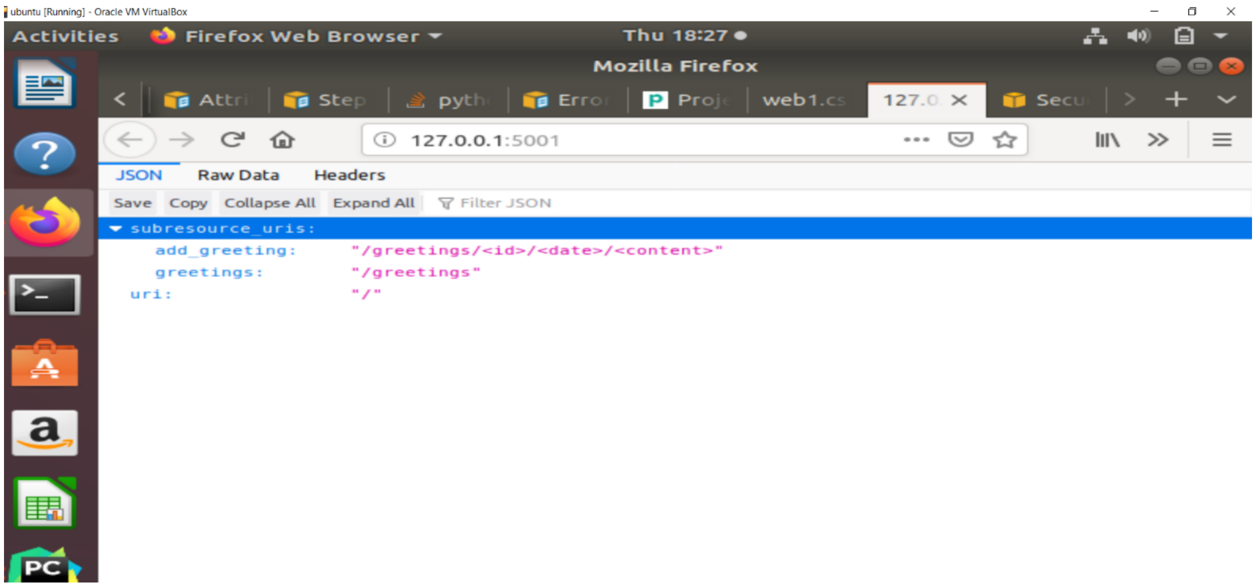
```
def read_table_item(table, pk_name, pk_value):
    """
    table is the object returned by get_table
    Return item read by primary key.
    """
    tabledata = get_table(table)
    response = tabledata.get_item(Key={pk_name: pk_value})
    return response


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)
```



```
subresource_uris:
  add_greeting:    "/greetings/<id>/<date>/<content>"
  greetings:       "/greetings"
uri:               "/"
```

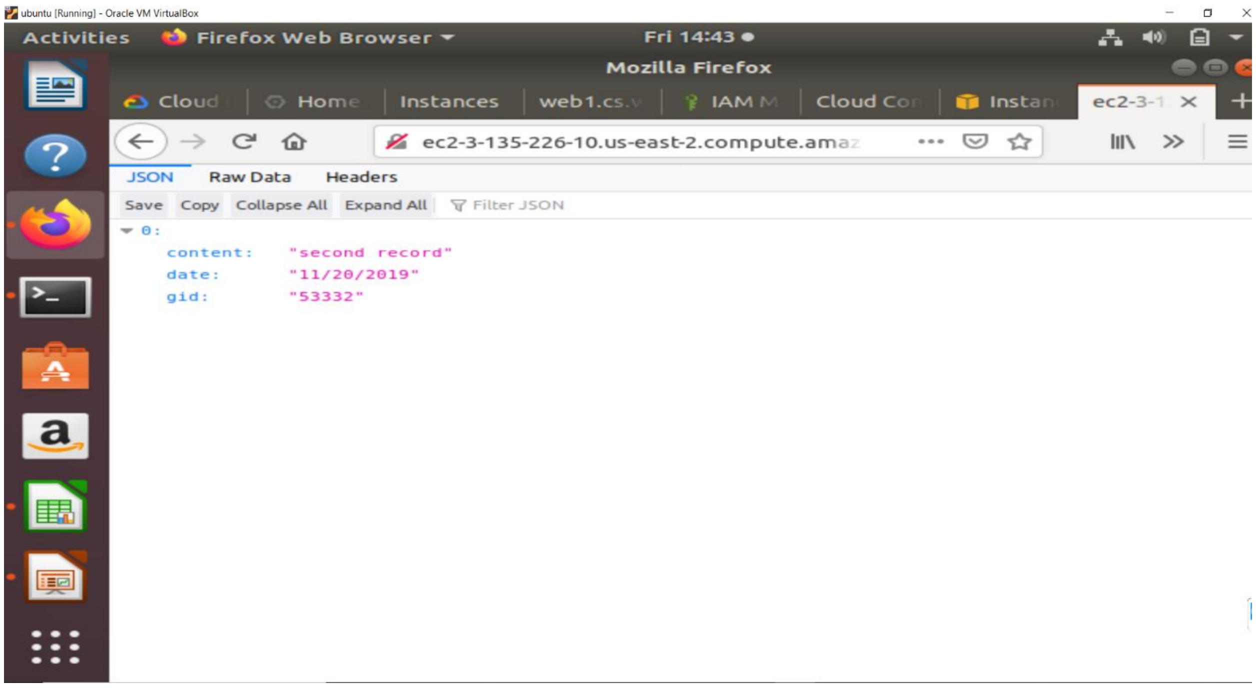
Deploying into Ec2 instance:

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.`

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**

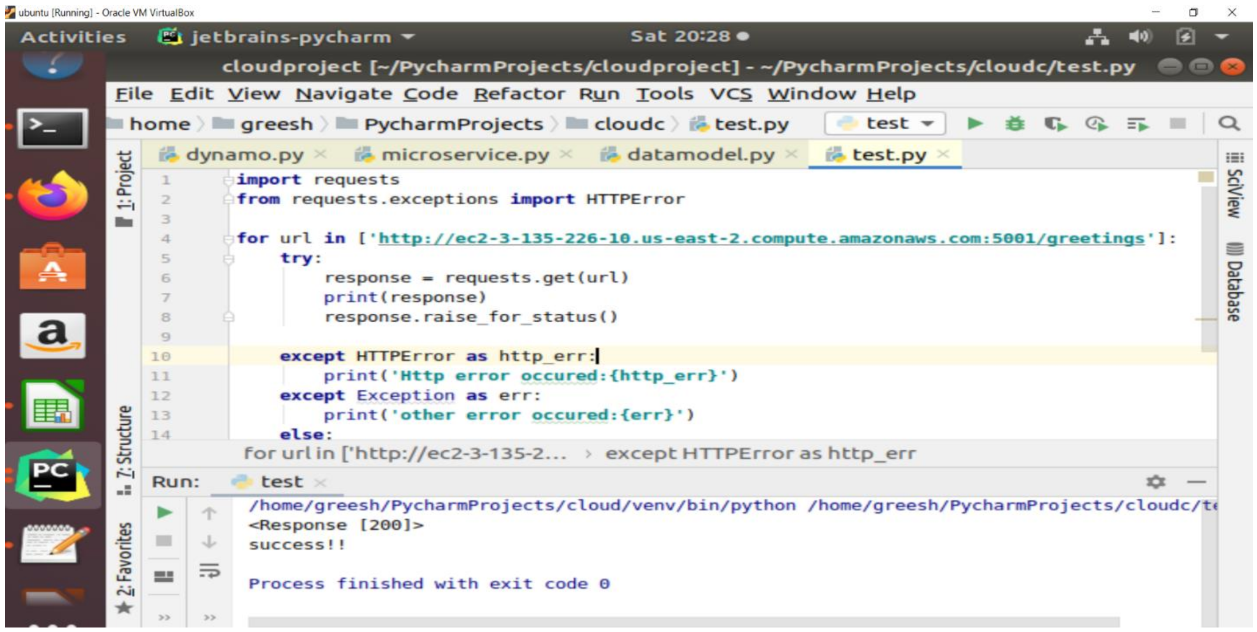


```
0:
    content:   "second record"
    date:      "11/20/2019"
    gid:       "53332"
```

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.


ubuntu [Running] - Oracle VM VirtualBox
Activities
jetbrains-pycharm
Sat 20:28
cloudproject [~/PycharmProjects/cloudproject] - ~/PycharmProjects/cloudc/test.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
home › greesh › PycharmProjects › cloudc › test.py
test
dynamo.py
microservice.py
datamodel.py
test.py
1: Project
7: Structure
2: Favorites
SciView
Database
Run:
test

```
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 occured:{http_err}')
    except Exception as err:
        print('other error occured:{err}')
    else:
```

for url in ['http://ec2-3-135-2... › except HTTPError as http_err

```
/home/greesh/PycharmProjects/cloud/venv/bin/python /home/greesh/PycharmProjects/cloudc/te
<Response [200]>
success!!

Process finished with exit code 0
```

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_requir
ed)
        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.