

ASSIGNMENT 3

Assignment Date	01-10-2022
Student Name	P.Sathya
Student Roll No	960519104073
Maximum Mark	2 Marks

1.Create a Bucket in IBM object storage.

Import Credentials

```
credentials = {  
    'IBM_API_KEY_ID': '*****',  
    'IAM_SERVICE_ID': '*****',  
    'ENDPOINT': '*****',  
    'IBM_AUTH_ENDPOINT': '*****',  
    'BUCKET': '*****',  
    'FILE': 'wine.csv'  
}  
  
cos = ibm_boto3.client(service_name='s3',  
    ibm_api_key_id=credentials['IBM_API_KEY_ID'],  
    ibm_service_instance_id=credentials['IAM_SERVICE_ID'],  
    ibm_auth_endpoint=credentials['IBM_AUTH_ENDPOINT'],  
    config=Config(signature_version='oauth'),  
    endpoint_url=credentials['ENDPOINT'])
```

File Uploads

```
# Upload file wine.csv from wine folder into project bucket as wine_data.csv  
cos.upload_file(Filename='wine/wine.csv',Bucket=credentials['BUCKET'],Key='wine_data.csv')
```

```

# upload zip file

cos.upload_file('wine.gz', credentials['BUCKET'],'wine.gz')

# Upload pickle object

cos.upload_file('GB_Classification_model.pkl', credentials['BUCKET'],'GB_Classification_model.pkl')

# upload file like object

with open('wine.csv', 'rb') as data:

    cos.upload_fileobj(data, credentials['BUCKET'], 'wine_bytes')

from ibm_botocore.client import Config

import ibm_boto3

```

```

def upload_file_cos(credentials,local_file_name,key):

    cos = ibm_boto3.client(service_name='s3',

    ibm_api_key_id=credentials['IBM_API_KEY_ID'],

    ibm_service_instance_id=credentials['IAM_SERVICE_ID'],

    ibm_auth_endpoint=credentials['IBM_AUTH_ENDPOINT'],

    config=Config(signature_version='oauth'),

    endpoint_url=credentials['ENDPOINT'])

    try:

        res=cos.upload_file(Filename=local_file_name, Bucket=credentials['BUCKET'],Key=key)

    except Exception as e:

        print(Exception, e)

    else:

        print(' File Uploaded')

```

```

upload_file_cos(credentials,'GB_Classification_model.pkl','GB_Classification_model1.pkl'

```

) File Uploaded

File Downloads

```

cos.download_file(Bucket=credentials['BUCKET'],Key='wine.csv',Filename='data/wine1.csv')

# download file like object

```

```

with open('wine_copy.csv', 'wb') as data:
    cos.download_fileobj(credentials['BUCKET'], 'wine_bytes', data)

from ibm_botocore.client import Config
import ibm_boto3

def download_file_cos(credentials,local_file_name,key):
    cos = ibm_boto3.client(service_name='s3',
        ibm_api_key_id=credentials['IBM_API_KEY_ID'],
        ibm_service_instance_id=credentials['IAM_SERVICE_ID'],
        ibm_auth_endpoint=credentials['IBM_AUTH_ENDPOINT'],
        config=Config(signature_version='oauth'),
        endpoint_url=credentials['ENDPOINT'])
    try:
        res=cos.download_file(Bucket=credentials['BUCKET'],Key=key,Filename=local_file_name)
    except Exception as e:
        print(Exception, e)
    else:
        print('File Downloaded')
download_file_cos(credentials,'model/GB_model.pkl','GB_Classification_model.pkl')
File Downloaded

```

New Credentials

```

cos_credentials={
    "apikey": "*****",
    "endpoints": "*****",
    "iam_apikey_description": "*****",
    "iam_apikey_name": "*****",
    "iam_role_crn": "*****",
    "iam_serviceid_crn": "*****",

```

```
"resource_instance_id": "*****"
}
```

```
auth_endpoint = 'https://iam.bluemix.net/oidc/token'
service_endpoint = 'https://s3-api.us-geo.objectstorage.softlayer.net'
```

```
cos = ibm_boto3.client('s3',
    ibm_api_key_id=cos_credentials['apikey'],
    ibm_service_instance_id=cos_credentials['resource_instance_id'],
    ibm_auth_endpoint=auth_endpoint,
    config=Config(signature_version='oauth'),
    endpoint_url=service_endpoint)
```

List Buckets

```
for bucket in cos.list_buckets()['Buckets']:
    print(bucket['Name'])
bluemixaccounts-hyx4v4raz-catalog-0422c6e2
buckettest
communitycosdf0fcb47bb7d48a1a847cee6cbe1bc57
cos-test-bucket1
cos-test-bucket2
cos1ab43f6f665aa4daaa9066513b83bdd32
cosproject062645eac3ca4746837c8897df3b7a0e
coswithoutenv2e0e51cec9bf472abaaf00aeebfdef7d
datacatalogandrefinetestc74f307cb1a74fec995e80d930357bac
demo9840d8da1d6049a8aa1da5e6906c41ee
dsx-sy8mm45a-catalog-0422c6e2
dsxenterpriseupsell0a087e0d42b24ba39ed1005696eec475
havi-r1hrlcyf-catalog-0422c6e2
```

havi914f33ced68240729566241410612716

music-bygusmcaz-catalog-0422c6e2

Create/Delete Buckets:

```
cos.create_bucket(Bucket='bucket1-test')
```

```
{'ResponseMetadata': {'HTTPHeaders': {'content-length': '0',  
'date': 'Tue, 30 Jan 2018 21:11:08 GMT',  
'server': 'Cleversafe/3.12.1.28',  
'x-amz-request-id': '6a8e444f-4ffa-4e0e-9f98-946df69ef346',  
'x-clv-request-id': '6a8e444f-4ffa-4e0e-9f98-946df69ef346',  
'x-clv-s3-version': '2.5'},  
'HTTPStatusCode': 200,  
'HostId': '',  
'RequestId': '6a8e444f-4ffa-4e0e-9f98-946df69ef346',  
'RetryAttempts': 0}}
```

```
cos.delete_bucket(Bucket='bucket1-test')
```

```
{'ResponseMetadata': {'HTTPHeaders': {'date': 'Tue, 30 Jan 2018 21:11:20  
GMT', 'server': 'Cleversafe/3.12.1.28',  
'x-amz-request-id': '631459c0-a70e-4492-83e3-52e2ff1e86b5',  
'x-clv-request-id': '631459c0-a70e-4492-83e3-52e2ff1e86b5',  
'x-clv-s3-version': '2.5'},  
'HTTPStatusCode': 204,  
'HostId': '',  
'RequestId': '631459c0-a70e-4492-83e3-52e2ff1e86b5',  
'RetryAttempts': 0}}
```

2.Upload an 5 images to the object storage and use the same page in your HTML code.

```
// upload to COS
```

```
await cos.upload({  
  Bucket: COS_BUCKET_NAME,  
  Key: `${fileDetails.userId}/${fileDetails.id}/${fileDetails.name}`,  
  Body: fs.createReadStream(file.path),  
  ContentType: fileDetails.type,  
}).promise();
```

Program:

```
<!DocType html>  
<html>  
  <head>  
    <title> Display Images </title>  
  </head>  
  <body>  
      
  </body>  
</html>
```

Output:



4.Design a chatbot using IBM Watson assistant for hospital.Ex.User comes with query to know the branches for that hospital in your city.Submit the web URL of that chatbot as a assignment.

Program:

```
import json
import logging
import os

logging.basicConfig(level=logging.INFO)
LOG = logging.getLogger(__name__)

default_name = 'insurance-voice-bot'
default_json = 'data/skill-insurance-voice-bot.json'
description = "Assistant workspace created by watson-voice-bot."

def init_skill(assistant_client):
```

```
"""Verify and/or initialize the Assistant workspace.
```

If a `WORKSPACE_ID` is specified in the runtime environment, make sure that workspace exists. If no `WORKSTATION_ID` is specified then try to find it using a lookup by name.

Name will be taken from the global `default_name` unless overridden using the `WORKSPACE_NAME` environment variable.

If a workspace is not found by ID or name, then try to create one from the JSON in file name specified by `default_json`. Use the name as mentioned above so future lookup will find what was created.

```
:param assistant_client: Assistant service client
```

```
:param object environ: runtime environment variables
```

```
:return: ID of Assistant workspace to use
```

```
:rtype: str
```

```
:raise Exception: When workspace is not found and cannot be created
```

```
"""
```

```
# Get the actual workspaces
```

```
workspaces = assistant_client.list_workspaces().get_result()[
```

```
'workspaces']
```

```
env_workspace_id = os.environ.get('WORKSPACE_ID')
```

```
if env_workspace_id:
```

```
# Optionally, we have an env var to give us a WORKSPACE_ID. #
```

```
If one was set in the env, require that it can be found.
```

```
LOG.info("Using WORKSPACE_ID=%s" % env_workspace_id) for
```


workspace in workspaces:

```
if workspace['workspace_id'] == env_workspace_id: ret =
```

```
env_workspace_id
```

```
break
```

```
else:
```

```
raise Exception("WORKSPACE_ID=%s is specified in a runtime "
```

```
"environment variable, but that workspace " "does not exist." %
```

```
env_workspace_id) else:
```

```
# Find it by name. We may have already created it.
```

```
name = os.environ.get('WORKSPACE_NAME', default_name) for
```

workspace in workspaces:

```
if workspace['name'] == name:
```

```
ret = workspace['workspace_id']
```

```
LOG.info("Found WORKSPACE_ID=%(id)s using lookup by "
```

```
"name=%(name)s" % {'id': ret, 'name': name}) break
```

```
else:
```

```
# Not found, so create it.
```

```
LOG.info("Creating workspace from " + default_json)
```

```
with open(default_json) as workspace_file:
```

```
workspace = json.load(workspace_file)
```

```
created = assistant_client.create_workspace(
```

```
name=name,
```

```
description=description,
```

```
language=workspace['language'],
```

```
metadata=workspace['metadata'],
```

```
intents=workspace['intents'],
```

```
entities=workspace['entities'],
```

```
dialog_nodes=workspace['dialog_nodes'],
```

```

counterexamples=workspace['counterexamples']).get_result()
ret = created['workspace_id']
LOG.info("Created WORKSPACE_ID=%(id)s with "
"name=%(name)s" % {'id': ret, 'name': name})
return ret

```

5.Create Watson assistant service with 10 steps and use 3 conditions in it.Load that script that in html page.

Program:

have the following empty page with the chatbot script so far :

```

<body style="height: 100%;">
<script src=https://assistant-web.watsonplatform.net/loadWatsonAssistantChat.js></script>
<script>
window.loadWatsonAssistantChat({
integrationID: "some id", // The ID of this integration.
region: "eu-gb" // The region your integration is hosted in.
}).then(function(instance){
instance.render();
});
</script>
</body>
</html>

```

```

<body style="height: 100%;">
<script src=https://assistant-web.watsonplatform.net/loadWatsonAssistantChat.js></script>
<script>
window.loadWatsonAssistantChat({

```

```

integrationID: "some id", // The ID of this integration.
region: "eu-gb", // The region your integration is hosted in.
options.openChatByDefault: true
}).then(function(instance){
instance.render();
});
</script>
</body>
</html>

```

```

window.watsonAssistantChatOptions = {
integrationID: "#####", // The ID of this integration.
region: "eu-gb", // The region your integration is hosted in.
serviceInstanceID: "#####", // The ID of your service instance.
onLoad: function(instance) { instance.render(); },
openChatByDefault: true
};
setTimeout(function(){
const t=document.createElement('script');
t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js"
document.head.appendChild(t);
});

```