ASSIGNMENT-3

Assignment Date	11/10/2022
Student Name	Jeffrey Benison A
Student Roll Number	95071914037
Maximum Marks	2 Marks

IBM OBJECT STORAGE:

#templates/index.html

```
<a href="/">HOME</a>
<a href="/uploader">Upload</a>
<a href="/deletefile">Delete</a>
<br>
<hr>
<h1>IBM Object Storage</h1>


<!doctype html>
<html>

<head>
   <link rel="stylesheet" href="static/style.css">
   <!-- href="{{ url_for('static',filename='style.css') }}"> -->
</head>

<body>
   {% for row in files %}
   <div style="border: 1px solid #EFEFEF;margin:10px;">
      <h3>Filename : {{row}} </h3>
      <img src="https://flask-test.s3.jp-tok.cloud-object-
storage.appdomain.cloud/{{row}}" width="150px"></td>
   </div>
   {% endfor %}
   <script>
      window.watsonAssistantChatOptions = {
          integrationID: "8294a8a4-9e3c-47d8-9478-515fb63886ef", // The ID of this
integration.
          region: "jp-tok", // The region your integration is hosted in.
          serviceInstanceID: "d065dad5-24d1-4292-b122-007bd6dffcad", // The ID of
your service instance.
          onLoad: function (instance) { instance.render(); }
       };
       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);
       });
    </script>
</body>

</html>
```

#templates/delete.html

```
<html>

<body>

    <a href="/">HOME</a>
    <a href="/uploader">Upload </a>
    <a href="/deletefile">Delete </a>
    <br>
    <hr>

    <h1>IBM Object Storage</h1>

    <form action="/deletefile" method="POST">
```

```
        <input type="text" placeholder="Enter bucket name"
name="bucket" />
        <br>
        <br>
        <input type="text" placeholder="Enter file name"
name="filename" />
        <br>
        <br>
        <input type="submit" />
    </form>
</body>

</html>
```

#templates/upload.html

```
<html>

<body>


        <a href="/">HOME</a>
        <a href="/uploader">Upload </a>
        <a href="/deletefile">Delete </a>
        <br>
        <hr>

        <h1>IBM Upload File</h1>
        <form action="/uploader" method="POST"
enctype="multipart/form-data">
                <input type="text" placeholder="Enter
bucket name" name="bucket" />
                <br>
                <br>
                <input type="text" placeholder="Enter file
name" name="filename" />
                <br>
                <br>
                <input type="file" name="file" />
                <br>
                <br>
                <input type="submit" />
        </form>
</body>

</html>
```

#static/style.css

```
* {
   background-color: grey;
}
```

#app.py

```
from flask import Flask, redirect, url_for,
render_template, request
import ibm_boto3
from ibm_botocore.client import Config, ClientError

COS_ENDPOINT = "https://s3.jp-tok.cloud-object-
storage.appdomain.cloud"
COS_API_KEY_ID = "DcQC8l1E_6PIq_bwHGHf-
_hmu95b11M-H6Qputp2VfjL"
COS_INSTANCE_CRN = "crn:v1:bluemix:public:cloud-
object-
storage:global:a/0834fd9d10254d12b564b9a26b86f44b:
802b5b02-ba01-491c-b67c-734e1f668dab::"


# Create resource https://s3.ap.cloud-object-
storage.appdomain.cloud
cos = ibm_boto3.resource("s3",
                    ibm_api_key_id=COS_API_KEY_ID,

ibm_service_instance_id=COS_INSTANCE_CRN,
                    config=Config(signature_version="oauth"),
                    endpoint_url=COS_ENDPOINT
                    )
```

```
app = Flask(__name__)


def get_item(bucket_name, item_name):
    print("Retrieving item from bucket: {0}, key:
{1}".format(
        bucket_name, item_name))
    try:
        file = cos.Object(bucket_name, item_name).get()

        print("File Contents: {0}".format(file["Body"].read()))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to retrieve file contents:
{0}".format(e))


def get_bucket_contents(bucket_name):
    print("Retrieving bucket contents from:
{0}".format(bucket_name))
    try:
        files = cos.Bucket(bucket_name).objects.all()
        files_names = []
        for file in files:
            files_names.append(file.key)
            print("Item: {0} ({1} bytes).".format(file.key,
file.size))
        return files_names
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to retrieve bucket contents:
{0}".format(e))


def delete_item(bucket_name, object_name):
    try:
        cos.Object(bucket_name, object_name).delete()
        print("Item: {0} deleted!\n".format(object_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to delete object: {0}".format(e))


def multi_part_upload(bucket_name, item_name,
file_path):
    try:
        print("Starting file transfer for {0} to bucket:
{1}\n".format(
            item_name, bucket_name))
        # set 5 MB chunks
        part_size = 1024 * 1024 * 5

        # set threadhold to 15 MB
        file_threshold = 1024 * 1024 * 15

        # set the transfer threshold and chunk size
        transfer_config =
ibm_boto3.s3.transfer.TransferConfig(
            multipart_threshold=file_threshold,
            multipart_chunksize=part_size
        )

        # the upload_fileobj method will automatically
execute a multi-part upload
        # in 5 MB chunks for all files over 15 MB
        with open(file_path, "rb") as file_data:
            cos.Object(bucket_name,
item_name).upload_fileobj(
                Fileobj=file_data,
                Config=transfer_config
            )

        print("Transfer for {0}
Complete!\n".format(item_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to complete multi-part upload:
{0}".format(e))


@app.route('/')
def index():
    files = get_bucket_contents('flask-test')
    return render_template('index.html', files=files)


@app.route('/deletefile', methods=['GET', 'POST'])
def deletefile():
    if request.method == 'POST':
        bucket = request.form['bucket']
        name_file = request.form['filename']

        delete_item(bucket, name_file)
        return 'file deleted successfully'

    if request.method == 'GET':
        return render_template('delete.html')


@app.route('/uploader', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        bucket = request.form['bucket']
        name_file = request.form['filename']
        f = request.files['file']
        multi_part_upload(bucket, name_file, f.filename)
        return 'file uploaded successfully <a href="/">GO to
Home</a>'

    if request.method == 'GET':
        return render_template('upload.html')


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)
```