Name: Deepak Thipeswamy

Course Number: CSE 6331

Programming Assignment- 1

The agenda is to host a web application on the IBM Bluemix cloud platform. Enable user to upload a file and the file is stored in a NoSQL DB (Cloudant) maintaining the different versions of the file uploaded. The user will also be able to download the file by providing the filename and the version he/she wants to download.

Requirements: Flask==0.10.1, requests, cloudant, yattag

Hosted at: <http://deepak1.mybluemix.net>

**welcome.py**

import os

import requests

from flask import Flask, request, send\_file

import pdb

import StringIO

from couchdbclient import \*

from time import gmtime, strftime

import hashlib

UPLOAD\_FOLDER = './uploads'

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

app.config['UPLOAD\_FOLDER'] = 'uploads/'

@app.route('/')

def Welcome():

return app.send\_static\_file('index.html')

@app.route('/upload', methods=['POST'])

def Uploaded():

# File uploaded will be handled here

fd = request.files['myfile']

if not fd:

return "No file"

docName = 'myFiles'

fname = fd.filename

fileData = fd.read()

hashValue = getHashValue(fileData)

curTime =strftime('%Y-%m-%d %H:%M:%S', gmtime())

#createDb()

createDocument(docName)

result = updateDocument(docName, fname, fileData, hashValue, curTime)

#deleteDocument(docName)

return result

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

def downloadOrDeleteFile():

# File download request will be handled here

version = int(request.form.get('Version'))

fname = str(request.form.get('Filename'))

if request.form['submit'] == 'Download':

print 'In download'

data = getMyFile(fname, 'myFiles', version)

if data == 'Not Found':

return 'File Not Found'

else:

print 'Got file'

strIO = StringIO.StringIO()

strIO.write(str(data))

strIO.seek(0)

return send\_file(strIO, attachment\_filename=fname, as\_attachment=True)

else:

return deleteFile(fname, version)

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

def listFiles():

# File download request will be handled here

return listMyFiles('myFiles')

def deleteFile(fname, version):

# File download request will be handled here

version = int(request.form.get('Version'))

fname = str(request.form.get('Filename'))

print 'In Delete File'

data = deleteMyFile('myFiles', fname, version)

return data

def getHashValue(fileData):

# Hashing of the file contents

hasher = hashlib.md5()

buf = fileData

hasher.update(buf)

print(hasher.hexdigest())

return str(hasher.hexdigest())

@app.route('/myapp')

def WelcomeToMyapp():

return 'Welcome again to my app running on Bluemix!'

port = os.getenv('VCAP\_APP\_PORT', '5000')

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

app.run(host='0.0.0.0', port=int(port))

**couchdbclient.py**

import json

import os

import requests

import pdb

from yattag import Doc

USERNAME = 'cad1d95f-9233-4933-9dfb-2a7472764e22-bluemix'

PASSWORD = '5d1b13d6c400a298a3a6301e5826b7da71a517fcb30343050d6689fbf8530461'

ACCOUNT\_NAME = 'my-cloudant'

creds = (USERNAME, PASSWORD)

baseURI = "https://{0}.cloudant.com/{1}".format(USERNAME, ACCOUNT\_NAME)

def createDb():

# Create Database

response = requests.put(

baseURI,

auth=creds

)

print "Created database at {0}".format(baseURI)

def createDocument(docName):

# Create a document on the database

response = requests.get(

"{0}/{1}".format(baseURI, docName),

auth=creds

)

# if document already present, ignore

if response.status\_code == 404:

response = requests.post(

baseURI,

data=json.dumps({

"\_id": docName,

"files": []

}),

auth=creds,

headers={"Content-Type": "application/json"}

)

docId = response.json()["id"]

print "The new document's ID is {0}".format(docId)

def updateDocument(docName, fname, fileData, hashValue, curTime):

# add data to the document

response = requests.get(

"{0}/{1}".format(baseURI, docName),

auth=creds

)

doc = response.json()

print "The document's rev is {0}".format(doc["\_rev"])

found = False

filesArray = doc['files']

max\_version = 0;

# Scan through all the files

for f in filesArray:

if str(f['filename']) == fname:

if max\_version < int(f['version\_number']):

max\_version = int(f['version\_number'])

if str(f['hashed\_value']) == hashValue:

found = True

break

else:

found = False

max\_version = max\_version + 1

# append to the existing list and increment version

if found == False:

doc['files'].append(dict(filename=fname,

version\_number=max\_version,

last\_modified\_date=curTime,

contents=fileData,

hashed\_value= hashValue,

))

else:

# Duplicate file found

return 'Duplicate File'

response = requests.put(

"{0}/{1}".format(baseURI, docName),

data=json.dumps(doc),

auth=creds

)

rev2 = response.json()['rev']

print "The document's new rev is {0}".format(rev2)

return 'File uploaded with version ' + str(max\_version)

def deleteMyFile(docName, fname, version):

# add data to the document

response = requests.get(

"{0}/{1}".format(baseURI, docName),

auth=creds

)

doc = response.json()

print "The document's rev is {0}".format(doc["\_rev"])

found = False

filesArray = doc['files']

# Scan through all the files

for f in filesArray:

if str(f['filename']) == fname:

if version == int(f['version\_number']):

found = True

filesArray.remove(f)

break

else:

found = False

if found == False:

return 'File not found'

else:

response = requests.put(

"{0}/{1}".format(baseURI, docName),

data=json.dumps(doc),

auth=creds

)

return 'File Deleted'

def getMyFile(filename, docName, version\_number):

# Download file request

response = requests.get(

"{0}/{1}".format(baseURI, docName),

auth=creds

)

doc = response.json()

filesArray = doc['files']

for f in filesArray:

if f['filename'] == filename:

if int(f['version\_number']) == version\_number:

return str(f['contents'])

return 'Not Found'

def listMyFiles(docName):

response = requests.get(

"{0}/{1}".format(baseURI, docName),

auth=creds

)

doc = response.json()

filesArray = doc['files']

# Scan through all the files

doc, tag, text = Doc().tagtext()

with tag('html'):

with tag('style'):text('table, th, td {border: 1px solid black; border-collapse: collapse;}th, td {padding: 5px;}')

with tag('body'):

with tag('table'):

with tag('tr', style="font-weight:bold"):

with tag('td'): text('Filename')

with tag('td'): text('Version')

with tag('td'): text('Last Modified On')

for f in filesArray:

with tag('tr'):

with tag('td'): text(str(f['filename']))

with tag('td'): text(str(f['version\_number']))

with tag('td'): text(str(f['last\_modified\_date']))

result = doc.getvalue()

return result

def deleteDocument(docName):

print "Deleting document"

response = requests.delete(

"{0}/{1}".format(baseURI, docName),

params={"rev": rev2},

auth=creds

)

print " > doc: ", response.json()

def deleteDatabase(baseURI, creds):

print 'Deleting database'

response = requests.delete(

baseURI,

auth=creds

)

print " > db: ", response.json()