QUESTIONS:

1. Flask-api-main
2. Flask-blog-with-db-main
3. Flask-with-ibm-cloud-object-storage-main
4. Flask-with-ibm-db2-main

Solution:

1. Flask-api-main

from flask import Flask, request app = Flask( name )

food\_items = { "1":"rice",

"2":"beans",

"3":"yam",

"4":"plantain",

"5":"potatoes",

"6":"wheat"

}

@app.route("/api") def index():

return "Hello form Flask API Server"

@app.route('/data', methods = ['POST', 'GET']) def api():

if request.method == 'GET': return food\_items

if request.method =='POST': data = request.json

food\_items.update(data) return "Data is inserted"

@app.route("/data/<id>", methods=["PUT"]) def update(id):

data = request.form['item'] food\_items[str(id)]=data return "Data updated"

@app.route("/data/<id>", methods=["DELETE"]) def delete(id):

food\_items.pop(str(id)) return "Data Deleted"

1. Flask-blog-with-db-main
   * post.html

<title>Posts</title>

<h2>Create New Blog Post:</h2>

<form action='/posts' method='POST'> Title: <br>

<input type='text' name='title' id='title' class="form- control">

<br> Author: <br>

<input type='text' name='author' id='author' class="form- control">

<br>

Post: <br>

<input type='text' name='content' id='content' class="form- control">

<br>

<input type='submit' value='Post' class="btn btn-success col-

sm-3">

</form>

<hr>

<h2>{{ post.title }}</h2>

<h3>By: {{ post.author }}</h3>

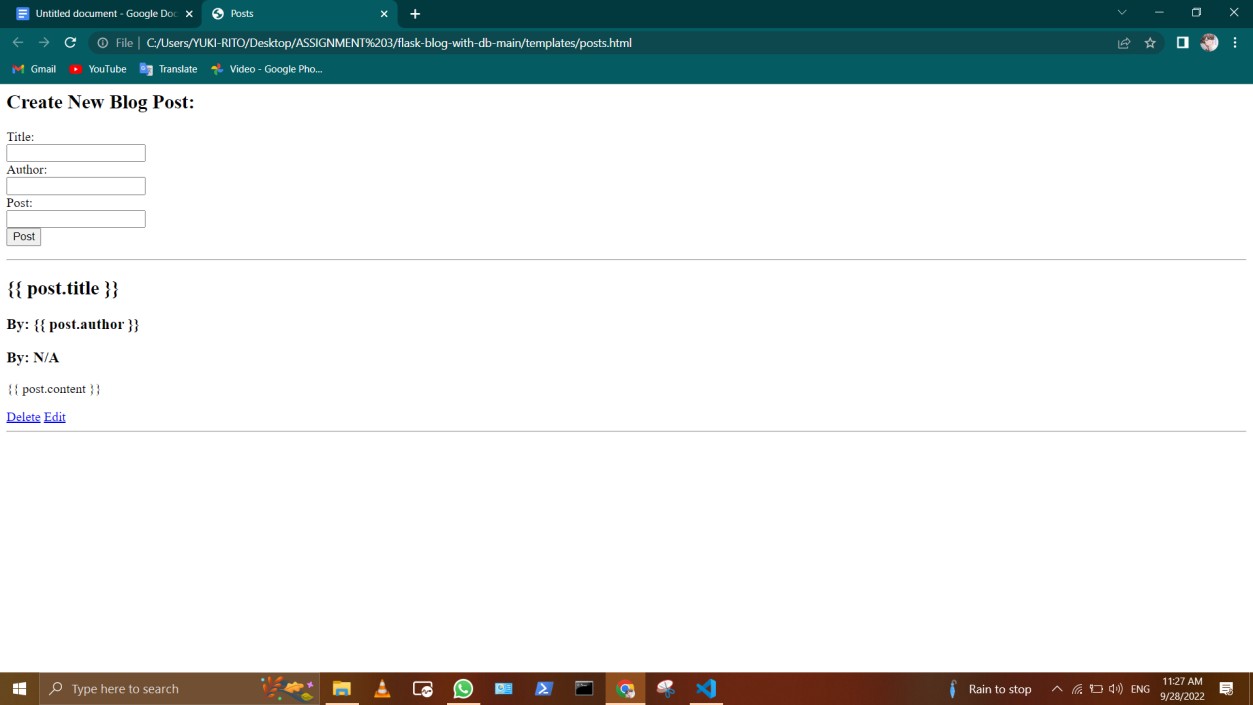
<h3>By: N/A</h3>

<p>{{ post.content }}</p>

<a href='/posts/delete/{{post.id}}'>Delete</a>

<a href='/posts/edit/{{post.id}}'>Edit</a>

<hr>

IMAGES:

* + Index.html

<title>Home</title>

<h1>Home Page</h1>

<hr>

<h2>{YUKI RITO}</h2>

<h3>By: {YUKI RITO.author }</h3>

<h3>By: N/A</h3>

<p>{YUKI RITO}</p>

<a href='/posts/delete/{{post.id}}'>Delete</a>

<a href='/posts/edit/{{post.id}}'>Edit</a>

<hr>

IMAGE:

* + EDIT.html

<title>Edit Post</title>

<h1>YUKI RITO</h1>

<hr>

<h2>Edit:</h2>

<form action='/posts/edit/{{post.id}}' method='POST'> Title: <input type='text' name='title' id='title'

value="{{post.title}}">

<br>

Author: <input type='text' name='author' id='author' value="{{post.author}}">

<br>

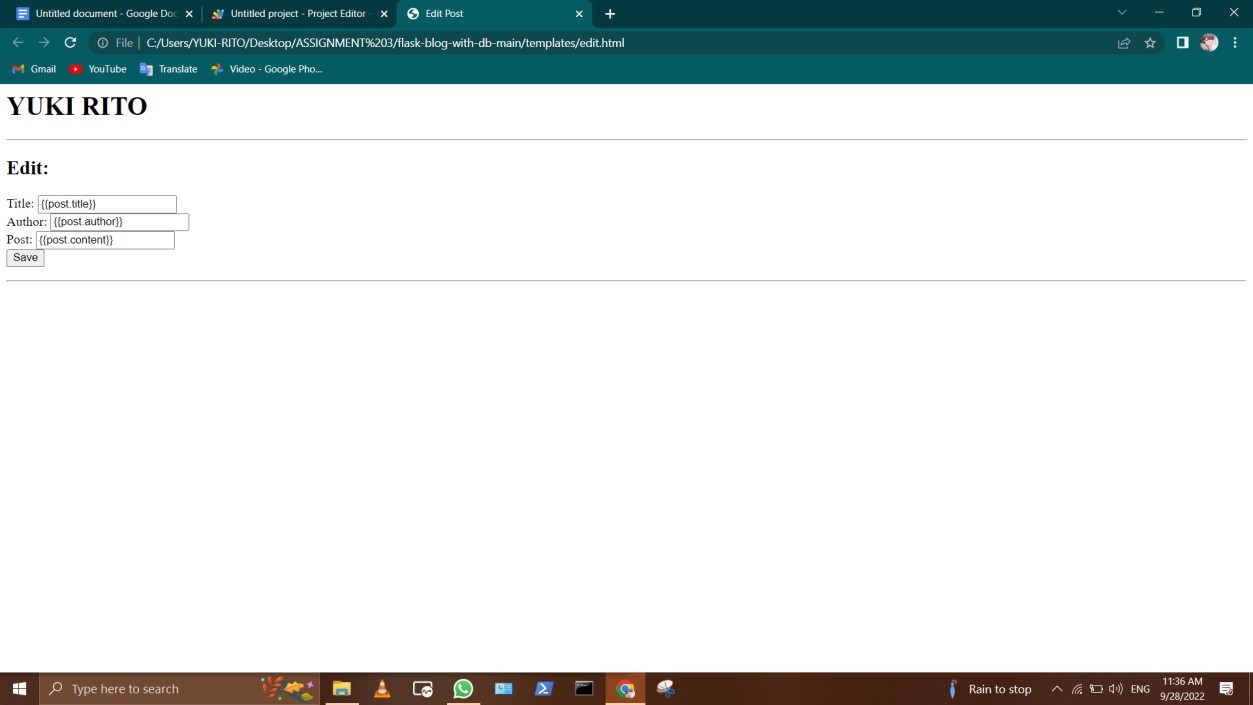
Post: <input type='text' name='content' id='content' value="{{post.content}}">

<br>

<input type='submit' value='Save'>

</form>

<hr>

IMAGE:

* + - Base.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial- scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<link href=["https://cdn.jsd](https://cdn.jsdelivr.net/npm/bootstrap%405.0.2/dist/css/bootstrap.m)e[livr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.m](https://cdn.jsdelivr.net/npm/bootstrap%405.0.2/dist/css/bootstrap.m) in.css" rel="stylesheet" integrity="sha384-

EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

<script src="[https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bun](https://cdn.jsdelivr.net/npm/bootstrap%405.0.2/dist/js/bootstrap.bun) dle.min.js" integrity="sha384- MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

<!-- <link rel="stylesheet" href="{{ url\_for('static', filename='css/main.css') }}"> -->

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<div class="container-fluid">

<a class="navbar-brand" href="/">Blog</a>

<button class="navbar-toggler" type="button" data-bs- toggle="collapse" data-bs-target="#navbarSupportedContent" aria- controls="navbarSupportedContent" aria-expanded="false" aria- label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="/">HOME</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/posts">NEW POST</a>

</li>

</ul>

<form class="d-flex">

<input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">

<button class="btn btn-outline-success" type="submit">Search</button>

</form>

</div>

</div>

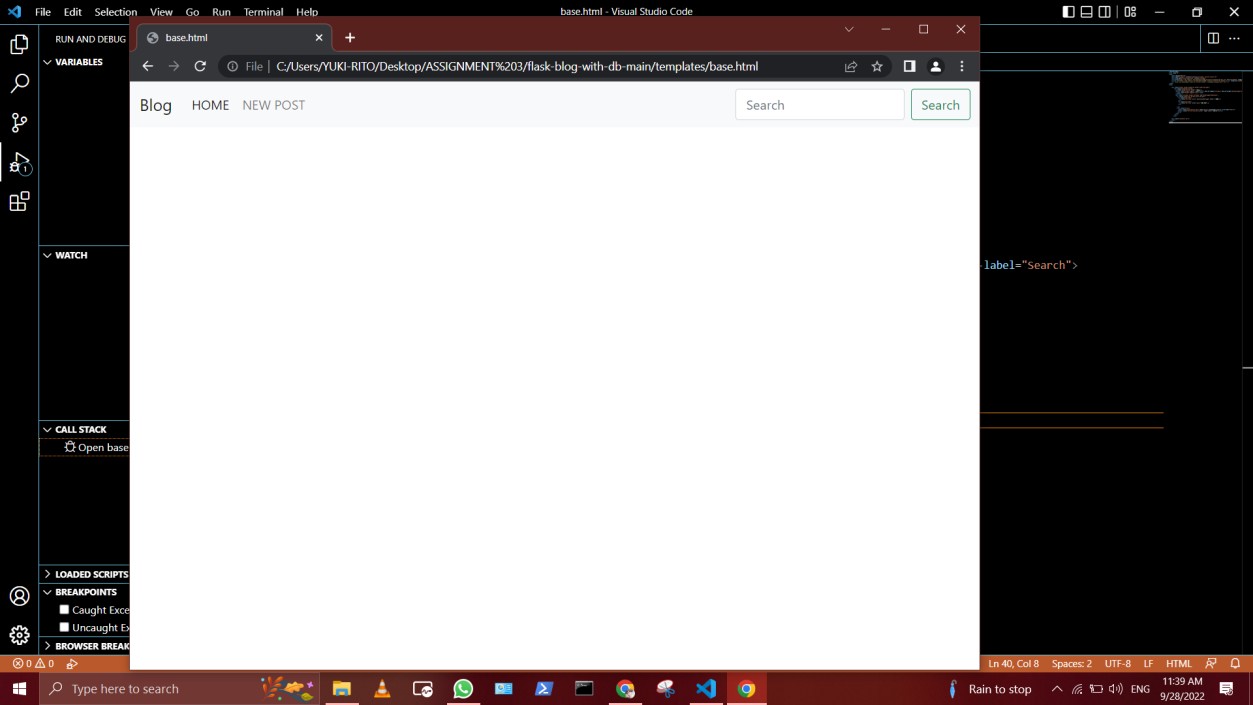
</nav>

<div class="container py-5">

</div>

</body>

</html>

IMAGE:

* + App.py

from flask import Flask, render\_template, request, redirect from flask\_sqlalchemy import SQLAlchemy

from datetime import datetime

app = Flask( name )

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///posts.db' db = SQLAlchemy(app)

class BlogPost(db.Model):

id = db.Column(db.Integer, primary\_key=True) title = db.Column(db.String(100), nullable=False) content = db.Column(db.Text, nullable=False)

author = db.Column(db.String(20), nullable=False, default='N/A') date\_posted = db.Column(db.DateTime, nullable=False,

default=datetime.utcnow)

def repr (self):

return 'Blog post ' + str(self.id)

@app.route('/')

def index():

all\_posts = BlogPost.query.order\_by(BlogPost.date\_posted).all() return render\_template('index.html', posts=all\_posts)

@app.route('/posts', methods=['GET', 'POST']) def posts():

if request.method == 'POST': post\_title = request.form['title']

post\_content = request.form['content'] post\_author = request.form['author']

new\_post = BlogPost(title=post\_title, content=post\_content, author=post\_author)

db.session.add(new\_post) db.session.commit() return redirect('/posts')

else:

all\_posts = BlogPost.query.order\_by(BlogPost.date\_posted).all() return render\_template('posts.html', posts=all\_posts)

@app.route('/posts/delete/<int:id>') def delete(id):

post = BlogPost.query.get\_or\_404(id) db.session.delete(post) db.session.commit()

return redirect('/posts')

@app.route('/posts/edit/<int:id>', methods=['GET', 'POST']) def edit(id):

post = BlogPost.query.get\_or\_404(id)

if request.method == 'POST': post.title = request.form['title']

post.author = request.form['author'] post.content = request.form['content'] db.session.commit()

return redirect('/posts') else:

return render\_template('edit.html', post=post)

if name == " main ":

app.run(debug=True)

1. Flask-with-ibm-cloud-object-storage-main
   * 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=" " COS\_INSTANCE\_CRN=""

# 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.delete\_object(Bucket=bucket\_name, Key=object\_name) 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('flaskapp123')

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)

* + - 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>

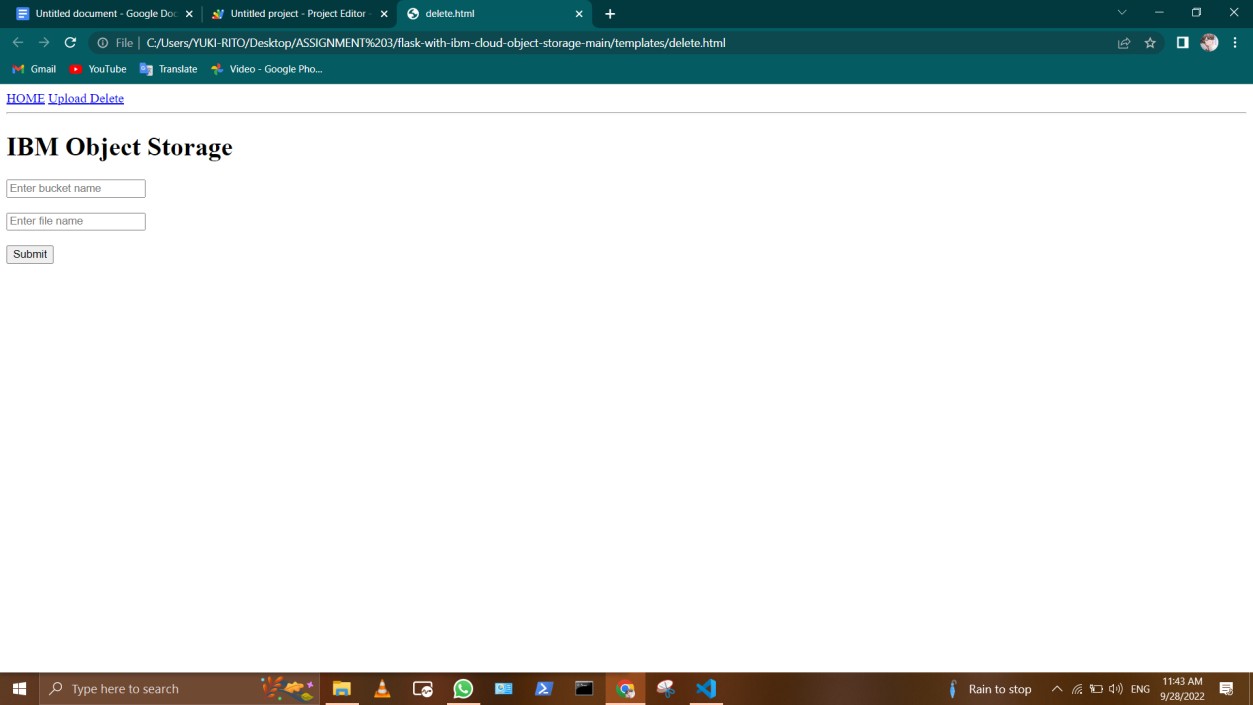
<input type = "submit"/>

</form>

</body>

</html>

IMAGE:



* + 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>

<body>

<div style="border: 1px solid #EFEFEF;margin:10px;">

<h3>Filename : {YUKI RITO} </h3>

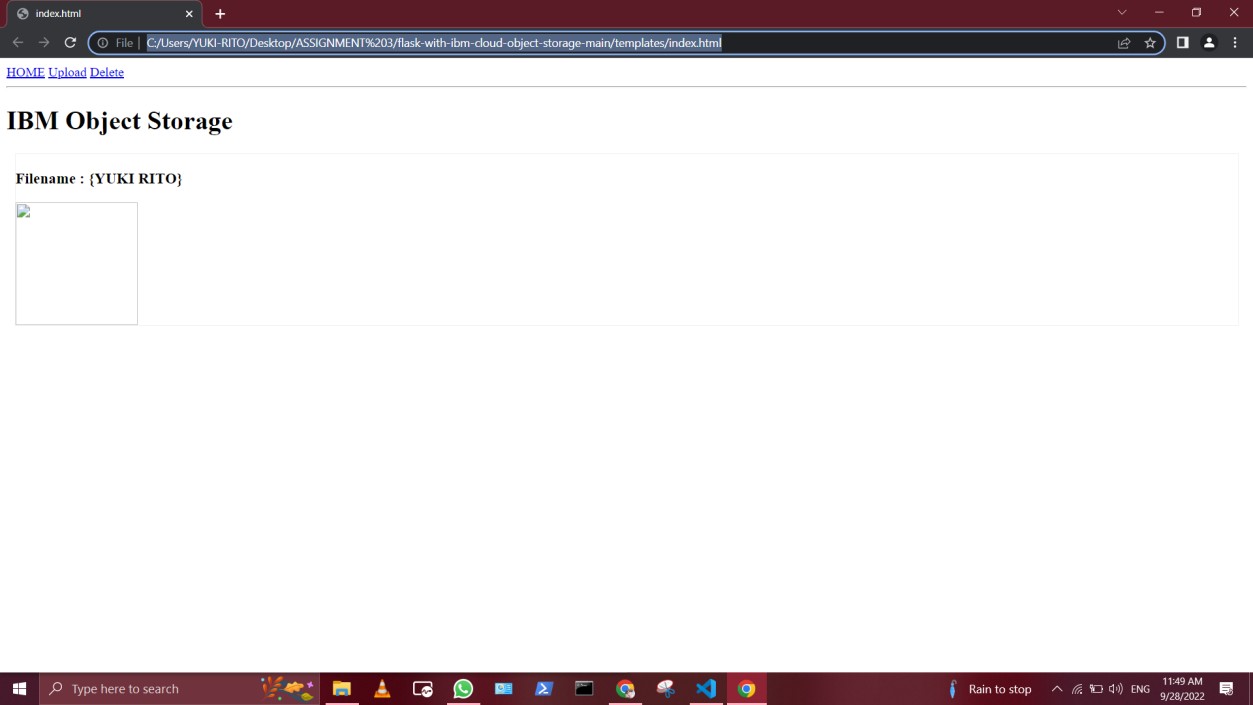
<img src="https://flaskapp123.s3.jp-tok.cloud-object- storage.appdomain.cloud/{{row}}" width="150px"></td>

</div>

</body>

</html>

Image:



* + 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>

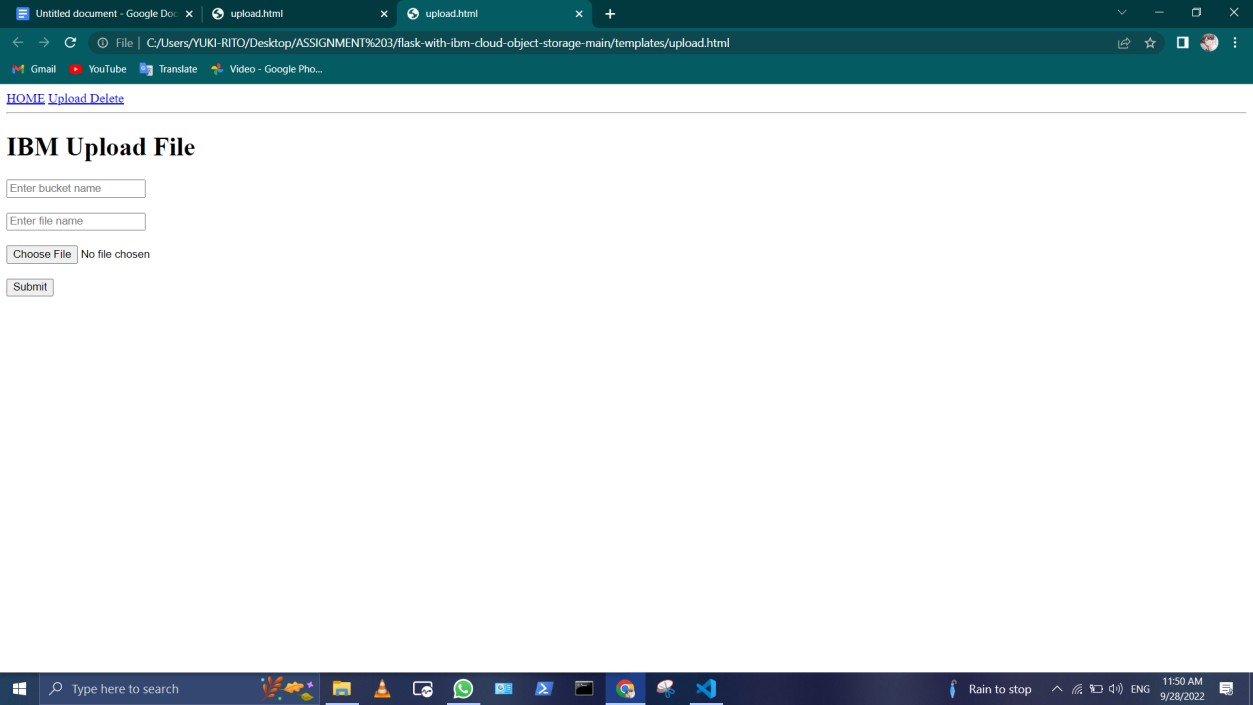
<br>

<input type = "submit"/>

</form>

</body>

</html>

IMAGE:

1. Flask-with-ibm-db2-main
   * APP.py

from turtle import st

from flask import Flask, render\_template, request, redirect, url\_for, session

from markupsafe import escape

import ibm\_db

conn = ibm\_db.connect("DATABASE=<databasename>;HOSTNAME=<your- hostname>;PORT=<portnumber>;SECURITY=SSL;SSLServerCertificate=DigiCertG lobalRootCA.crt;UID=<username>;PWD=<password>",'','')

app = Flask( name )

@app.route('/') def home():

return render\_template('home.html')

@app.route('/addstudent') def new\_student():

return render\_template('add\_student.html')

@app.route('/addrec',methods = ['POST', 'GET']) def addrec():

if request.method == 'POST':

name = request.form['name'] address = request.form['address'] city = request.form['city']

pin = request.form['pin']

sql = "SELECT \* FROM students WHERE name =?" stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,name) ibm\_db.execute(stmt)

account = ibm\_db.fetch\_assoc(stmt)

if account:

return render\_template('list.html', msg="You are already a member, please login using your details")

else:

insert\_sql = "INSERT INTO students VALUES (?,?,?,?)" prep\_stmt = ibm\_db.prepare(conn, insert\_sql) ibm\_db.bind\_param(prep\_stmt, 1, name)

ibm\_db.bind\_param(prep\_stmt, 2, address)

ibm\_db.bind\_param(prep\_stmt, 3, city)

ibm\_db.bind\_param(prep\_stmt, 4, pin) ibm\_db.execute(prep\_stmt)

return render\_template('home.html', msg="Student Data saved successfuly..")

@app.route('/list') def list():

students = []

sql = "SELECT \* FROM Students"

stmt = ibm\_db.exec\_immediate(conn, sql) dictionary = ibm\_db.fetch\_both(stmt) while dictionary != False:

# print ("The Name is : ", dictionary)

students.append(dictionary) dictionary = ibm\_db.fetch\_both(stmt)

if students:

return render\_template("list.html", students = students)

@app.route('/delete/<name>') def delete(name):

sql = f"SELECT \* FROM Students WHERE name='{escape(name)}'" print(sql)

stmt = ibm\_db.exec\_immediate(conn, sql) student = ibm\_db.fetch\_row(stmt)

print ("The Name is : ", student) if student:

sql = f"DELETE FROM Students WHERE name='{escape(name)}'" print(sql)

stmt = ibm\_db.exec\_immediate(conn, sql)

students = []

sql = "SELECT \* FROM Students"

stmt = ibm\_db.exec\_immediate(conn, sql) dictionary = ibm\_db.fetch\_both(stmt) while dictionary != False:

students.append(dictionary) dictionary = ibm\_db.fetch\_both(stmt)

if students:

return render\_template("list.html", students = students, msg="Delete successfully")

# # while student != False:

# # print ("The Name is : ", student)

# print(student) return "success..."

# @app.route('/posts/edit/<int:id>', methods=['GET', 'POST']) # def edit(id):

# post = BlogPost.query.get\_or\_404(id)

# if request.method == 'POST':

# # # # # #

#

post.title = request.form['title'] post.author = request.form['author'] post.content = request.form['content'] db.session.commit()

return redirect('/posts') else:

return render\_template('edit.html', post=post)

* + ADD\_STUDENT .HTML

<a href="/">HOME</a>

<a href="/addstudent">Add New Student</a>

<a href="/list">List Student</a>

<hr>

<form action = "{{ url\_for('addrec') }}" method = "POST">

<h3>Student Information</h3> Name<br>

<input type = "text" name="name" /></br>

Address<br>

<textarea name="address" ></textarea><br>

City<br>

<input type = "text" name="city" /><br>

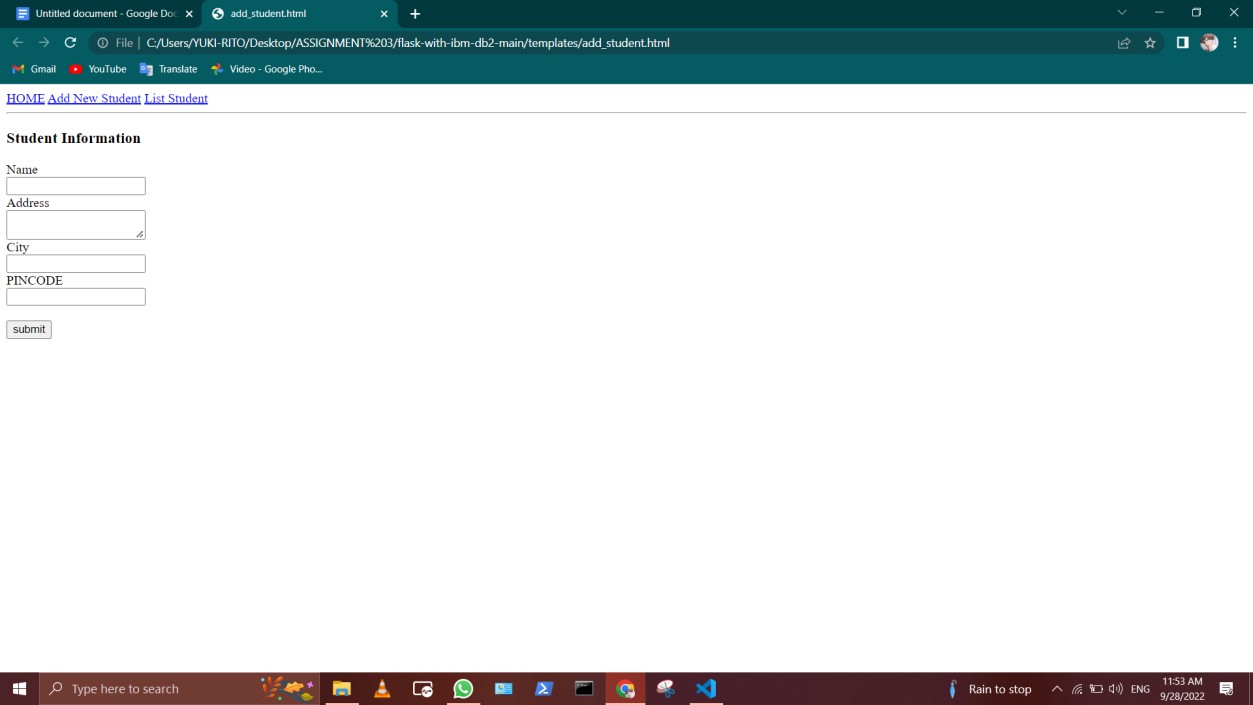
PINCODE<br>

<input type = "text" name="pin" /><br><br>

<input type = "submit" value = "submit" /><br>

</form>

IMAGE:



* + HOME.HTML

●

●

●

●

●

●

●

●

●

<a href="/">HOME</a>

<a href="/addstudent">Add New Student</a>

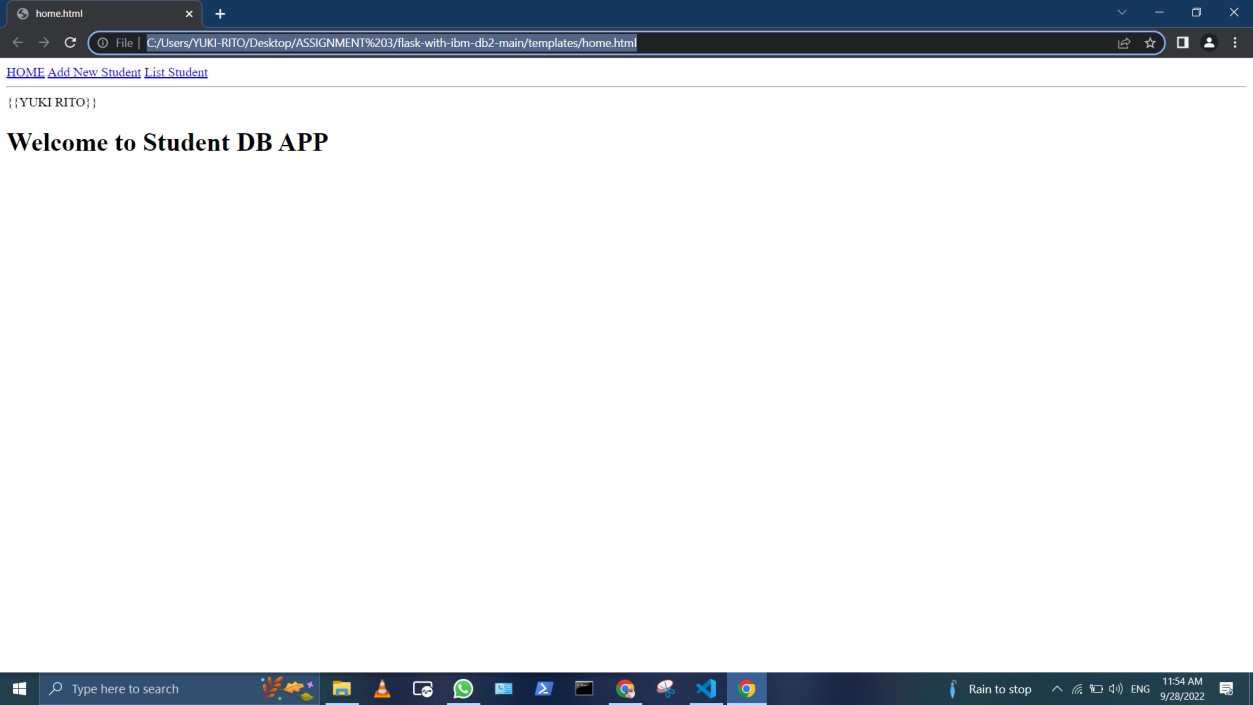
<a href="/list">List Student</a>

<hr>

{{YUKI RITO}}

<h1>Welcome to Student DB APP</h1>

IMAGE:



* + - List.html

<!doctype html>

<html>

<body>

<a href="/">HOME</a>

<a href="/addstudent">Add New Student</a>

<a href="/list">List Student</a>

<br><hr>

{DATA ENTRY}

<table border = 1>

<thead>

<td>Name</td>

<td>Address</td>

<td>city</td>

<td>Pincode</td>

<td></td>

</thead>

<tr>

<td>YUKI RITO</td>

<td>ANAICUT</td>

<td>VELLORE</td>

<td>632101</td>

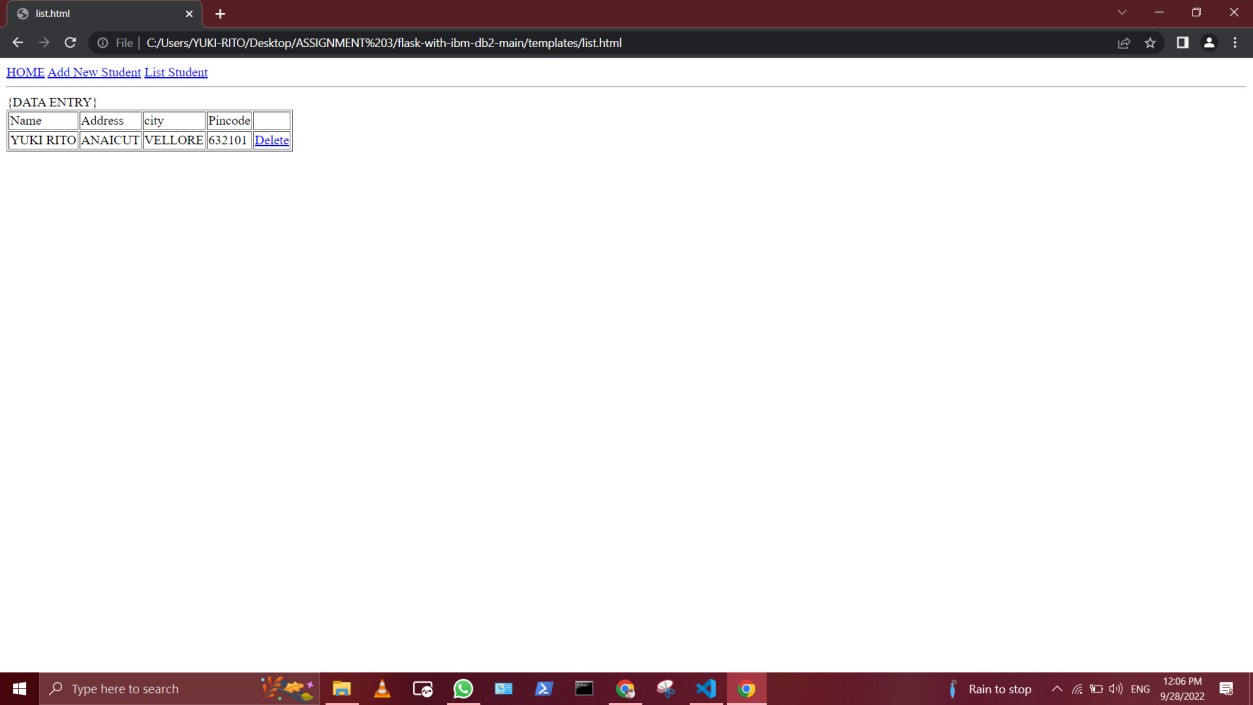
<td><a href="/delete/{{row['NAME']}}">Delete</a></td>

</tr>

</table>

</body>

</html>

Image:

* + RESULT.html

<!doctype html>

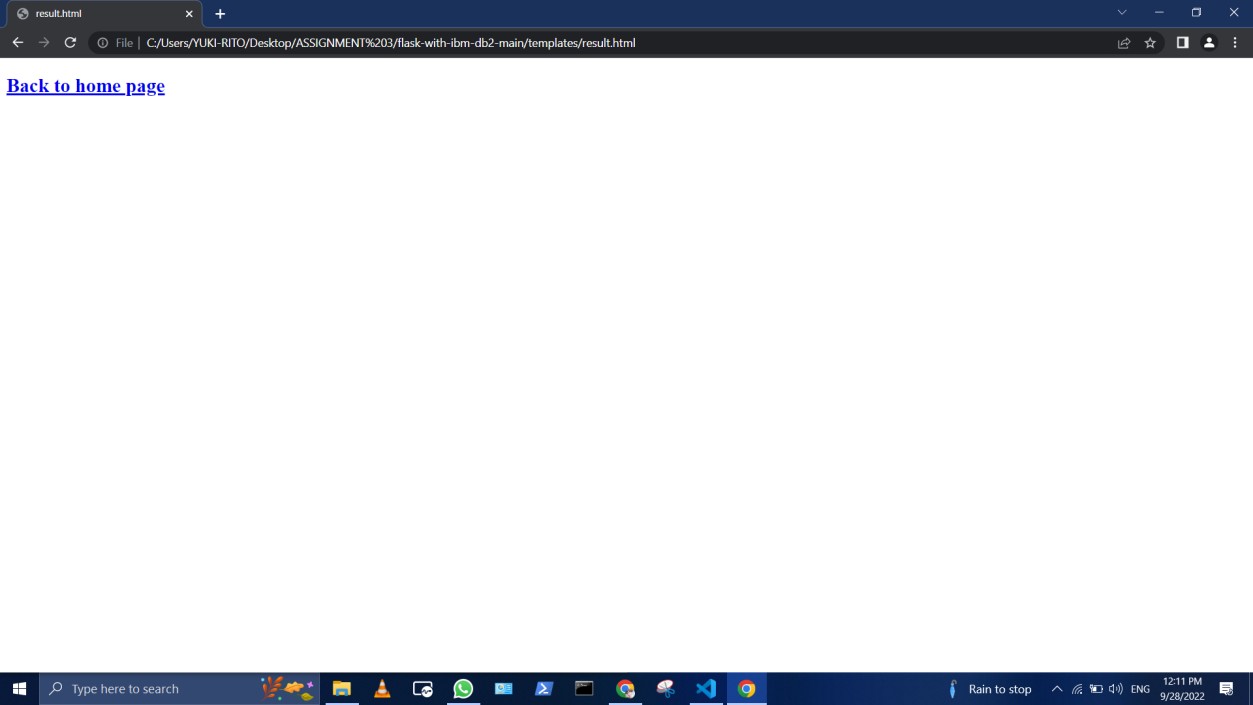
<html>

<body>

<h2><a href = "\">Back to home page</a></h2>

</body>

</html>

IMAGE: