

Faculty of Information Technology

SUBJECT NAME: ADVANCED DESIGN PATTERN
SUBJECT CODE: ADP631

I declare that I am familiar with, and will abide to the Examination rules of CTU


Signature

Summative Assessment

Duration:

Date:

Total Marks: 300

Total pages: 15

Examiner: Mr. Gerhard L

Moderator: Mr. Isaac L

Student number

| | | | | | | | | |
|---|---|---|---|--|--|--|--|--|
| 7 | 2 | 9 | 9 | | | | | |
|---|---|---|---|--|--|--|--|--|

Surname:

Van Straaten

Initials:

R

/

%

Student Name: Ruan
Student Surname: Van Straaten
Campus Name: Bloemfontein

ADP631_SA Section-2 Evidence

Contents

| | |
|---|----|
| PART-1 SERVERLESS | 4 |
| Question-1.1 | 4 |
| 1.1 | 4 |
| 1.2 | 4 |
| Question-1.2 | 5 |
| 2.1 | 5 |
| 2.2 | 5 |
| 2.3 Supply a screenshot of the load_data Lambda function's execution logs in CloudWatch. | 7 |
| 3.2 Supply AWS portal screenshots of your ADP631GateWay. | 10 |
| 3.3 Supply a screenshot of your get_weather function's IAM Role's policies. | 10 |
| 3.4 Supply a screenshot of your get_weather function's json output to the client (browser). ... | 11 |
| 3.5 Supply a screenshot of your get_weather function's execution logs in CloudWatch. | 12 |
| Question-1.4 | 13 |
| 4.1 Supply a screenshot of your website-123 S3 bucket. containing the weather.js file (amongst the other static files). | 13 |
| 4.2 Supply a text copy of your weather.js file, containing your Ajax function. | 14 |
| 4.3 Supply a screenshot of your index.html page displaying a city with its weather details. | 16 |
| PART-2 CONTAINERS..... | 17 |
| Question-2.1 | 17 |
| 1.1. | 17 |
| 1.2. Supply a screenshot of your containers running in the Docker Dashboard. | 18 |
| 1.3. Supply a screenshot of your browser connecting to RabbitMQ in the container. | 19 |
| 1.4. Supply a screenshot of SSMS connecting to the SQL Server in the container. | 19 |
| Question-2.2 | 20 |
| 2.1. Controller Action..... | 20 |
| 2.2. | 21 |
| 2.3. | 21 |
| 2.5. Supply text copies of your startup.cs file in the Products Service. | 22 |
| 2.7. Supply screenshots of successful and invalid requests and responses in the client (e.g. Postman) pertaining to the Products Service. | 24 |
| 2.8. Supply text copies of any DTO's or other classes not already included in the Products Service..... | 25 |
| Question-2.3 | 26 |
| 3.1. Supply text copies of any Costing Service controller actions..... | 26 |
| 3.2. Supply text copies of any Costing Service background services and any other services that you might have used to consume events/messages from the Ordered queue on RabbitMQ..... | 27 |

| | |
|--|----|
| 3.3. | 28 |
| 3.4. | 28 |
| 3.5. Supply text copies of your startup.cs file in the Costing Service. | 29 |
| 3.6. Supply text copies of your launchSettings.json file in the Costing Service..... | 30 |
| 3.7. Supply screenshots of your Cost table's records displayed in SSMS (Total_Amount should not be a column in the table)..... | 31 |
| 3.8. Supply screenshots of the Costing service's HTML output (Total_Amount should be included in the HTML table). | 32 |
| 3.9. Supply text copies of your model and context classes. | 33 |
| 3.10. Supply text copies of any DTO's or other classes not already included. | 34 |

PART-1 SERVERLESS

Question-1.1

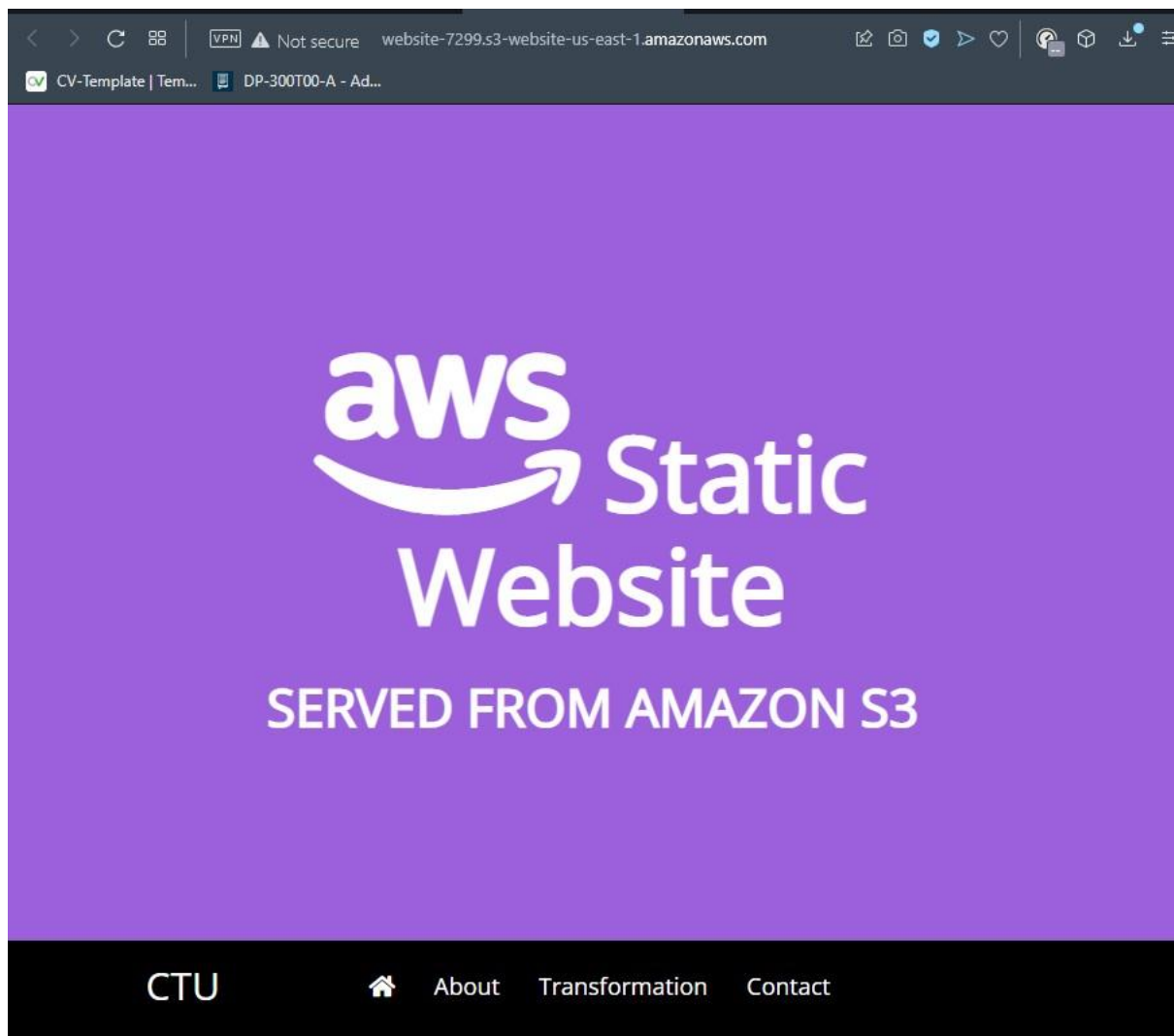
1.1

Files and folders (4 Total, 16.0 KB)

Find by name:

| Name | Folder | Type | Size | Status | Error |
|------------|--------|-----------------|--------|-------------|-------|
| error.html | - | text/html | 3.2 KB | ✓ Succeeded | - |
| index.html | - | text/html | 7.5 KB | ✓ Succeeded | - |
| script.js | - | text/javascript | 2.0 KB | ✓ Succeeded | - |
| style.css | - | text/css | 3.3 KB | ✓ Succeeded | - |

1.2



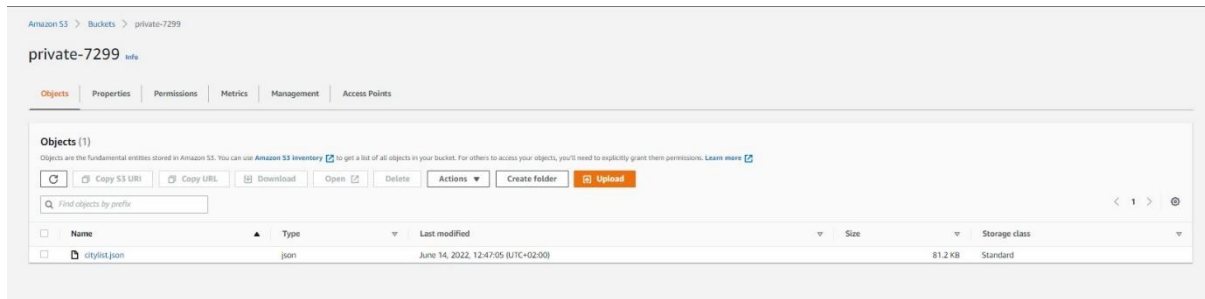
ABOUT US

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec nisl odio.

Mauris vehicula at nunc id posuere.

Question-1.2

2.1



2.2

import json

import boto3

import ast

from decimal import Decimal

```
s3_client = boto3.client('s3')
```

```
dynamodb = boto3.resource('dynamodb')
```

```
def lambda_handler(event, context):
```

```
    print(json.dumps(event))
```

```
    #fetch bucket name
```

```
    bucket = event['Records'][0]['s3']['bucket']['name']
```

```
    #fetch the file name which is uploaded
```

```
    file_name = event['Records'][0]['s3']['object']['key']
```

```
    #read the file
```

```
    json_object = s3_client.get_object(Bucket= bucket ,Key = file_name )
```

```
    file_reader = json_object['Body'].read().decode('utf-8')
```

```
    #converts string to dictionary to push to dynamodb
```

```
    file_reader = ast.literal_eval(file_reader)
```

```
    #converts all the float to decimal values for further processing
```

```
    file_reader = json.loads(json.dumps(file_reader), parse_float=Decimal)
```

```
    #temp dict to save to data base
```

```
    pushToDB = {}
```

```
    table = dynamodb.Table('valid_cities')
```

```
    for node in file_reader['city_list']:
```

```
        try:
```

```
            #save values in separate dict ID / NAME / Country
```

```
            pushToDB["id"] = node['id']
```

```
            pushToDB["name"] = node['name']
```

```

pushToDB["country"] = node['country']
#push values to the dynamodb
table.put_item(Item=pushToDB)
except Exception as e:
    print(e)
print ('Saved to db')

```

The screenshot shows a code editor with a menu bar (File, Edit, Find, View, Go, Tools, Window) and buttons for 'Test' and 'Deploy'. A status bar at the top right indicates 'Changes not deployed'. The left sidebar shows the 'Environment' with a folder 'load_data - /' and a file 'lambda_function.py'. The main editor area shows the code for 'lambda_function.py' with line numbers 1 through 40. The code imports json, boto3, ast, and Decimal from decimal. It initializes s3_client and dynamodb. The lambda_handler function prints the event, fetches bucket and file names, reads the file, converts it to a dictionary, and pushes it to a DynamoDB table named 'valid_cities'. It includes error handling for exceptions.

```

1 import json
2 import boto3
3 import ast
4 from decimal import Decimal
5
6
7 s3_client = boto3.client('s3')
8 dynamodb = boto3.resource('dynamodb')
9
10 def lambda_handler(event, context):
11     print(json.dumps(event))
12     #fetch bucket name
13     bucket = event['Records'][0]['s3']['bucket']['name']
14
15     #fetch the file name which is uploaded
16     file_name = event['Records'][0]['s3']['object']['key']
17
18     #read the file
19     json_object = s3_client.get_object(Bucket= bucket ,Key = file_name )
20     file_reader = json_object['Body'].read().decode('utf-8')
21
22     #converts string to dictionary to push to dynamodb
23     file_reader = ast.literal_eval(file_reader)
24     #converts all the float to decimal values for further processing
25     file_reader = json.loads(json.dumps(file_reader), parse_float=Decimal)
26
27     #temp dict to save to data base
28     pushToDB = {}
29     table = dynamodb.Table('valid_cities')
30     for node in file_reader['city_list']:
31         try:
32             #save values in separate dict ID / NAME / Country
33             pushToDB["id"] = node['id']
34             pushToDB["name"] = node['name']
35             pushToDB["country"] = node['country']
36             #push values to the dynamodb
37             table.put_item(Item=pushToDB)
38         except Exception as e:
39             print(e)
40     print ('Saved to db')

```

2.3 Supply a screenshot of the load_data Lambda function's execution logs in CloudWatch.

Log streams (100+) 🔄 Delete Create log stream Se...

By default, we only load the most recent log streams. [Load more](#).

< 1 2 ...

| <input type="checkbox"/> | Log stream | ▼ | Last event time |
|--------------------------|--|---|---------------------------------|
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]84e3111a02b247a2bc1bbd3906474cbc | | 2022-06-16 12:33:17 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]edc25f1a5ff541fbac4a31bb9264ee4e | | 2022-06-16 12:28:21 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]ec99fd5632c64ca48c397d85080f8e1c | | 2022-06-16 12:24:09 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]ccb6eaf387f6494db55cdd384aff9c1a | | 2022-06-16 12:23:39 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]3131194ab7fe4de497b94d0f37595a15 | | 2022-06-16 12:22:36 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]f5c8243de5f64992aab5f41a8e00fb27 | | 2022-06-16 12:18:55 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]63d3eaffef1d4d03b01423f237faee2d | | 2022-06-16 12:17:14 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]29d44c9687b341429a65e53470a85e19 | | 2022-06-16 12:14:44 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]43b41ffde344451ab2556fb451baabc2 | | 2022-06-16 12:08:31 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]58b8e7952b554b58bdcec4f490cee6aa | | 2022-06-16 12:07:42 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]c8b88103f5f74fd8be88315728449410 | | 2022-06-16 12:06:24 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]1efbe3e68f5e426c9a4b800e2e44548c | | 2022-06-16 12:05:57 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]ea8f0c5bb45740d9a554653076701fbd | | 2022-06-16 12:05:07 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]00ee43448b5340a1adaccf3a95446014 | | 2022-06-16 12:04:25 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]0a9d33e488ab4ab3b4a694ca0265d61d | | 2022-06-16 12:02:41 (UTC+02:00) |
| <input type="checkbox"/> | 2022/06/16/[\$LATEST]f5cceff167e884ce0816a26f4d43e93cb | | 2022-06-16 12:01:53 (UTC+02:00) |

Items returned (50) 🔄 Actions ▼ Create item

< 1 ... > ⚙️ 🔍

| <input type="checkbox"/> | name | ▼ | country | ▼ | id | ▼ |
|--------------------------|------------------|---|---------|---|---------|---|
| <input type="checkbox"/> | Eersterus | | ZA | | 1105774 | |
| <input type="checkbox"/> | Rietvlei | | ZA | | 961331 | |
| <input type="checkbox"/> | Orania | | ZA | | 967542 | |
| <input type="checkbox"/> | eSikhawini | | ZA | | 1005029 | |
| <input type="checkbox"/> | Generaal de Wet | | ZA | | 1002174 | |
| <input type="checkbox"/> | Reservoir Hills | | ZA | | 962649 | |
| <input type="checkbox"/> | Port Saint Johns | | ZA | | 964408 | |
| <input type="checkbox"/> | Erand | | ZA | | 1005265 | |
| <input type="checkbox"/> | Vanadjou | | KM | | 1092041 | |
| <input type="checkbox"/> | Bophelong | | ZA | | 1017447 | |
| <input type="checkbox"/> | Ebenhaezer | | ZA | | 1006919 | |
| <input type="checkbox"/> | Bloemfontein | | ZA | | 1018725 | |
| <input type="checkbox"/> | Onverwacht | | ZA | | 967827 | |
| <input type="checkbox"/> | Standerton | | ZA | | 952747 | |
| <input type="checkbox"/> | Dundee | | ZA | | 1007400 | |
| <input type="checkbox"/> | Josefsdeel | | ZA | | 993643 | |
| <input type="checkbox"/> | Meersig | | NA | | 3364236 | |

Question-1.3

3.1

```
import json
import boto3
from boto3.dynamodb.conditions import Key
import webbrowser
from urllib.request import urlopen
import urllib, json
```

```
def lambda_handler(event, context):
    client = boto3.resource('dynamodb')
    table = client.Table('valid_cities')

    #grabs city name and country from the api
    city_name = event['params']['path']['name']
    city_country = event['params']['querystring']['country']

    #get item by name in the database to check if city and country code exists in database
    try:
        response=table.get_item(
            Key={
                'name': city_name,
                'country': city_country
            }
        )

        #openweathermap.org request
        url = "http://api.openweathermap.org/data/2.5/weather?q=" + city_name + ","
        +city_country + "&APPID=80ea139dbc19d7a8a7a01874d540be4b"

        #save the json respond inside a dict
        response = urllib.request.urlopen(url)
        data = json.loads(response.read())

        #converts kelvin to celsius
        #temp
        temp = data['main']['temp']
        temp = temp - 273.15
        format_temp = "{:.2f}".format(temp)
        data['main']['temp'] = format_temp

        #min temp
        min_temp = data['main']['temp_min']
        min_temp = min_temp - 273.15
        format_min = "{:.2f}".format(min_temp)
```

```
data['main']['temp_min'] = format_min
```

```
#max temp
```

```
max_temp = data['main']['temp_max']
```

```
max_temp = max_temp - 273.15
```

```
format_max = "{:.2f}".format(max_temp)
```

```
data['main']['temp_max'] = format_max
```

```
#feels like temp
```

```
feels_like = data['main']['feels_like']
```

```
feels_like = feels_like - 273.15
```

```
format_feel = "{:.2f}".format(feels_like)
```

```
data['main']['feels_like'] = format_feel
```

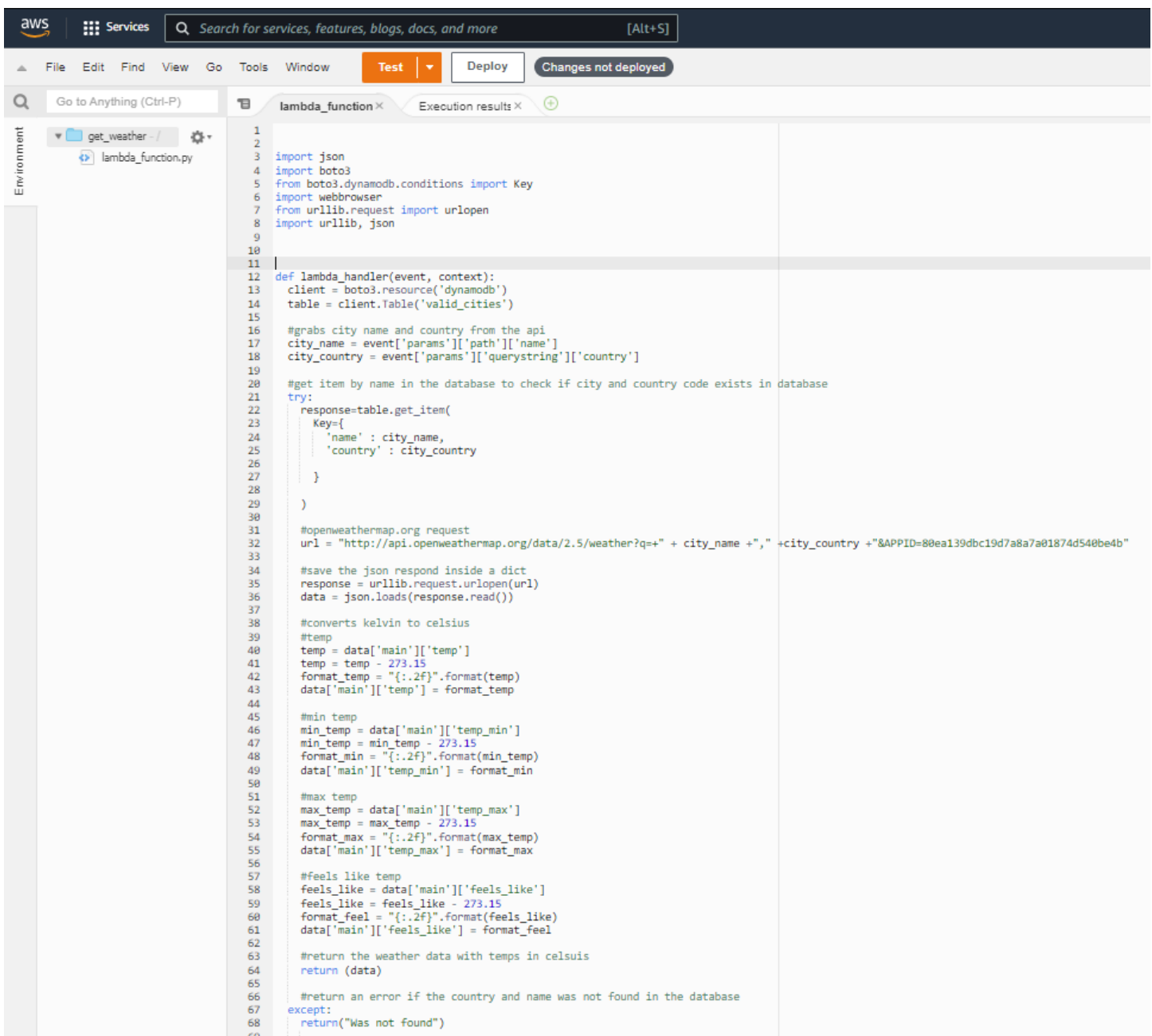
```
#return the weather data with temps in celsuis
```

```
return (data)
```

```
#return an error if the country and name was not found in the database
```

```
except:
```

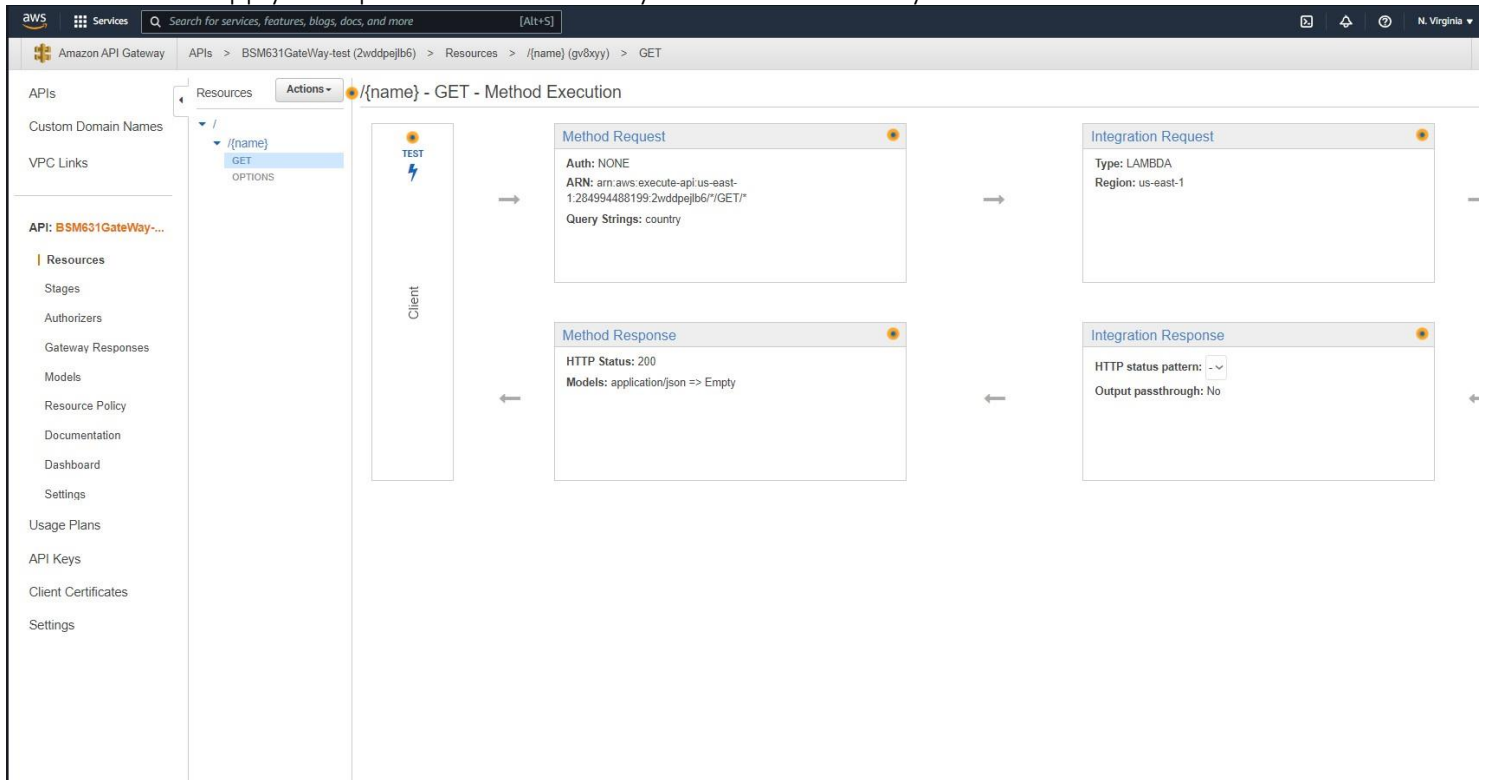
```
return("Was not found")
```



The screenshot shows the AWS Lambda console interface. At the top, there's a search bar and navigation tabs for 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test', and 'Deploy'. Below the tabs, there's a 'Go to Anything (Ctrl-P)' search bar. On the left, the 'Environment' pane shows a file tree with 'get_weather - /' and 'lambda_function.py'. The main area displays the code for 'lambda_function.py' with line numbers 1 through 69. The code is a Python function that interacts with a DynamoDB database to fetch weather data from the OpenWeatherMap API. It includes comments in German explaining each step, from database lookup to API request, data conversion from Kelvin to Celsius, and formatting the final JSON response. The function also includes an exception handler to return an error message if the city and country are not found in the database.

```
1
2
3 import json
4 import boto3
5 from boto3.dynamodb.conditions import Key
6 import webbrowser
7 from urllib.request import urlopen
8 import urllib, json
9
10
11
12 def lambda_handler(event, context):
13     client = boto3.resource('dynamodb')
14     table = client.Table('valid_cities')
15
16     #grabs city name and country from the api
17     city_name = event['params']['path']['name']
18     city_country = event['params']['queryString']['country']
19
20     #get item by name in the database to check if city and country code exists in database
21     try:
22         response=table.get_item(
23             Key={
24                 'name' : city_name,
25                 'country' : city_country
26             }
27         )
28
29     )
30
31     #openweathermap.org request
32     url = "http://api.openweathermap.org/data/2.5/weather?q=" + city_name + "," + city_country + "&APPID=80ea139dbc19d7a8a7a01874d540be4b"
33
34     #save the json respond inside a dict
35     response = urllib.request.urlopen(url)
36     data = json.loads(response.read())
37
38     #converts kelvin to celsius
39     #temp
40     temp = data['main']['temp']
41     temp = temp - 273.15
42     format_temp = "{:.2f}".format(temp)
43     data['main']['temp'] = format_temp
44
45     #min temp
46     min_temp = data['main']['temp_min']
47     min_temp = min_temp - 273.15
48     format_min = "{:.2f}".format(min_temp)
49     data['main']['temp_min'] = format_min
50
51     #max temp
52     max_temp = data['main']['temp_max']
53     max_temp = max_temp - 273.15
54     format_max = "{:.2f}".format(max_temp)
55     data['main']['temp_max'] = format_max
56
57     #feels like temp
58     feels_like = data['main']['feels_like']
59     feels_like = feels_like - 273.15
60     format_feel = "{:.2f}".format(feels_like)
61     data['main']['feels_like'] = format_feel
62
63     #return the weather data with temps in celsuis
64     return (data)
65
66     #return an error if the country and name was not found in the database
67 except:
68     return("Was not found")
69
```

3.2 Supply AWS portal screenshots of your ADP631GateWay.



3.3 Supply a screenshot of your get_weather function's IAM Role's policies.

get_weather-role-nihr505o

Summary

Creation date

June 16, 2022, 14:48 (UTC+02:00)

Last activity

✓ 22 minutes ago

ARN

am:aws:iam::284994488199:role/service-role/get_weather-role-nihr505o

Maximum session duration

1 hour

Permissions

Trust relationships

Tags

Access Advisor

Revoke sessions

Permissions policies (3)

You can attach up to 10 managed policies.



Simulate

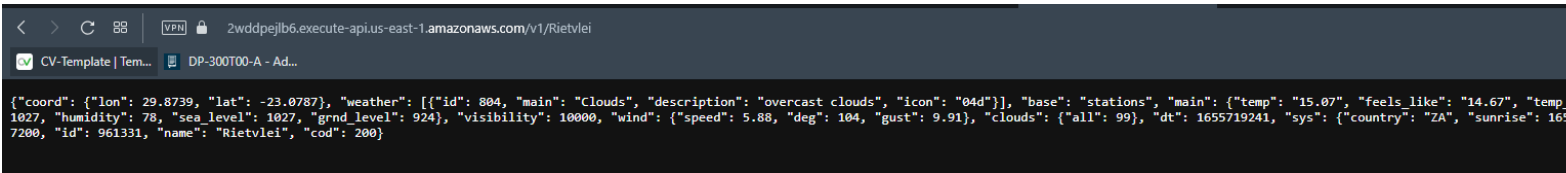
Remove

Add p

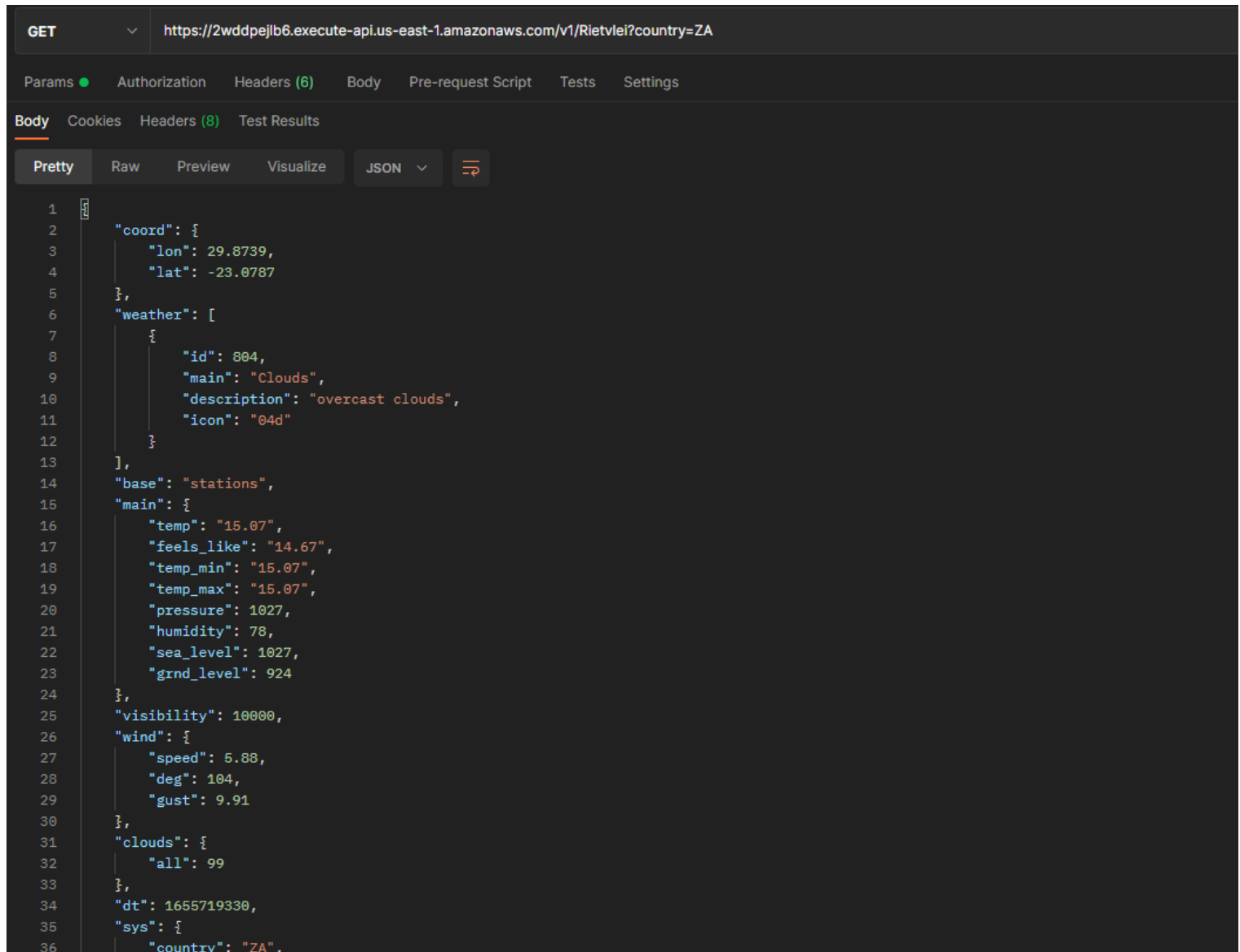
Filter policies by property or policy name and press enter

| <input type="checkbox"/> | Policy name | Type | Description |
|--------------------------|--|------------------|-------------|
| <input type="checkbox"/> | AWSLambdaBasicExecutionRole-1e889b52-0023-4071-84d7-b8372deade3e | Customer managed | |
| <input type="checkbox"/> | AmazonS3FullAccess | AWS managed | Provides f |
| <input type="checkbox"/> | AmazonDynamoDBFullAccess | AWS managed | Provides f |





3.4 Supply a screenshot of your get_weather function's json output to the client (browser).



In Postman for better viewing



3.5 Supply a screenshot of your get_weather function's execution logs in CloudWatch.

| Log streams (96+) | | |  |  |  |  |
|---|---|--|---|---|---|---|
| By default, we only load the most recent log streams. Load more. | | | | | | |
| <input type="text" value="Filter loaded log streams or try prefix search"/> | | | < 1 2 ... > @ | | | |
| <input type="checkbox"/> | Log stream | | Last event time | | | |
| <input type="checkbox"/> | 2022/06/19/[\${LATEST}]1fb55bde270a4ed4b940435f27e991d3 | | 2022-06-19 12:51:04 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/19/[\${LATEST}]fac68e6090f24b8482140e9c45f9a3a2 | | 2022-06-19 12:44:47 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/19/[\${LATEST}]250da1a3c4ac48709d2a14b21e50b8a0 | | 2022-06-19 12:18:42 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/19/[\${LATEST}]0f92dec3f7541acae7ddb06f8763b0e | | 2022-06-19 11:31:17 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]32365e0c089b498a864e4cfe63f2f9b0 | | 2022-06-18 17:06:44 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]1cb8c2359ce2489eaa67bd25d5e56e62 | | 2022-06-18 16:57:12 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]aa4dca327ff42c991ecb5de99705910 | | 2022-06-18 16:47:19 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]5d9729a1de9b4d52864cf8d01629d333 | | 2022-06-18 16:35:46 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]84a9094cb44a4e5e854ee5a8e9daa2a | | 2022-06-18 15:53:15 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]925969abb15f418582ce9fe0658bb21 | | 2022-06-18 15:46:11 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]18caffcb5df945b6ba6b6dca23f8e6 | | 2022-06-18 15:31:52 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]033264283bb54e6ab8ba46118d6d8e69 | | 2022-06-18 15:18:38 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]c40d9f7dd92d4390a4bf99f836a5e066 | | 2022-06-18 15:05:32 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]3a5a5edfe0904b29b7c9cce06601ad10 | | 2022-06-18 14:46:21 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]3bbf1412b12b4f19d1a25f0e9ef2caa | | 2022-06-18 11:57:19 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]eccc4a0a664b4dd1bfef04c6b61103cd | | 2022-06-18 11:56:01 (UTC+02:00) | | | |
| <input type="checkbox"/> | 2022/06/18/[\${LATEST}]09b564fd88134789a5de60ccd766fb84 | | 2022-06-18 11:55:45 (UTC+02:00) | | | |

Question-1.4

4.1 Supply a screenshot of your website-123 S3 bucket. containing the weather.js file (amongst the other static files).

Objects (5)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

| <input type="checkbox"/> | Name | Type | Last modified | Size |
|--------------------------|----------------------------|------|-------------------------------------|------|
| <input type="checkbox"/> | error.html | html | June 14, 2022, 10:14:53 (UTC+02:00) | |
| <input type="checkbox"/> | index.html | html | June 19, 2022, 12:50:40 (UTC+02:00) | |
| <input type="checkbox"/> | script.js | js | June 14, 2022, 10:14:52 (UTC+02:00) | |
| <input type="checkbox"/> | style.css | css | June 14, 2022, 10:14:50 (UTC+02:00) | |
| <input type="checkbox"/> | weather.js | js | June 19, 2022, 12:50:39 (UTC+02:00) | |

4.2 Supply a text copy of your weather.js file, containing your Ajax function.

```
async function getWet()
{
    //Get data from text boxes
    var name = document.getElementById('fname').value;
    var country = document.getElementById('fcountry').value;
    console.log(name + country)

    try {
        //send request to the api
        const URL = ('https://2wddpejlb6.execute-api.us-east-1.amazonaws.com/v1/' + name + '?country=' + country)
        //gets the response in json format
        const response = await fetch(URL);
        const data = await response.json();

        //prints the response inside the html page
        var output = document.getElementById('displayhere')
        output.innerHTML =
        "City Info: " + "<br>" +
        "City name: " + data.name + "<br>" +
        "Country Code: " + data.sys['country'] + "<br>" +
        "Lon : " + data.coord['lon'] + "<br>" +
        "Lat : " + data.coord['lat'] + "<br>" + "<br>" +

        "General Weather Info:" + "<br>" +
        "temperature is: " + data.main['temp'] + " Degree Celcuis" + "<br>" +
        "Minum Temperature: " + data.main['temp_min'] + " Degree Celcuis" +
        "<br>" +
        "Max Temperature: " + data.main['temp_max'] + " Degree Celcuis" +
        "<br>" +
        "Feels like: " + data.main['feels_like'] + " Degree Celcuis" +
        "<br>" +
        "Humidity : " + data.main['humidity'] + "<br>" +
        "Pressure : " + data.main['pressure'] + "<br>" +
        "Sea Level : " + data.main['sea_level'] + "<br>" +
        "Ground Level : " + data.main['grnd_level'] + "<br>" +
        "Sky : " + data.weather[0]["description"] + "<br>" + "<br>" +
        "Wind Info: " + "<br>" +
        "Wind Speed : " + data.wind['speed'] + "km/h" + "<br>" +
        "Gust Speed : " + data.wind['gust'] + "km/h" + "<br>" +
        "Wind Deg : " + data.wind['deg'] + "<br>"
        //in case of error
    } catch (error) {
        var output = document.getElementById('displayhere')
```

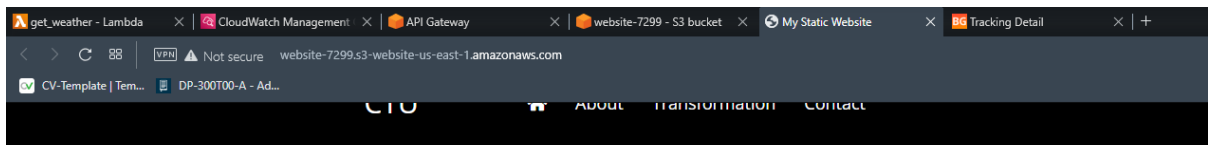
<> index.html

JS weather.js ●

JS weather.js > getWet

```
1  async function getWet()
2  {
3      //Get data from text boxes
4      var name = document.getElementById('fname').value;
5      var country = document.getElementById('fcountry').value;
6      console.log(name + country)
7
8      try {
9          //send request to the api
10         const URL = ('https://2wddpejlb6.execute-api.us-east-1.amazonaws.com/v1/'+name+'?country='+country)
11         //gets the response in json format
12         const response = await fetch(URL);
13         const data = await response.json();
14
15         //prints the response inside the html page
16         var output = document.getElementById('displayhere')
17         output.innerHTML =
18         "City Info: " + "<br>" +
19         "City name: " + data.name + "<br>" +
20         "Country Code: " + data.sys['country'] + "<br>" +
21         "Lon : " + data.coord['lon'] + "<br>" +
22         "Lat : " + data.coord['lat'] + "<br>" + "<br>" +
23
24         "General Weather Info:" + "<br>" +
25         "Temperature is: " + data.main['temp'] + " Degree Celcuis" + "<br>" +
26         "Minum Temprature: " + data.main['temp_min'] + " Degree Celcuis" + "<br>" +
27         "Max Temperature: " + data.main['temp_max'] + " Degree Celcuis" + "<br>" +
28         "Feels like: " + data.main['feels_like'] + " Degree Celcuis" + "<br>" +
29         "Humidity : " + data.main['humidity'] + "<br>" +
30         "Pressure : " + data.main['pressure'] + "<br>" +
31         "Sea Level : " + data.main['sea_level'] + "<br>" +
32         "Ground Level : " + data.main['grnd_level'] + "<br>" +
33         "Sky : " + data.weather[0]['description'] + "<br>" + "<br>" +
34
35         "Wind Info: " + "<br>" +
36         "Wind Speed : " + data.wind['speed'] + "km/h" + "<br>" +
37         "Gust Speed : " + data.wind['gust'] + "km/h" + "<br>" +
38         "Wind Deg : " + data.wind['deg'] + "<br>"
39
40
41
42         //in case of error
43         catch (error) {
44             var output = document.getElementById('displayhere')
45             output.innerHTML = "Invalid Reqeust"
46         }
47     }
48 }
```


4.3 Supply a screenshot of your index.html page displaying a city with its weather details.



ABOUT US

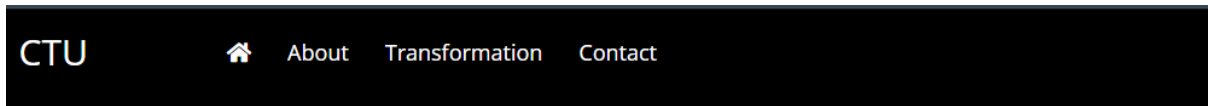
Name Of The City :

Country Code:

DATA

City Info:
City name: Rietvlei
Country Code: ZA
Lon : 29.8739
Lat : -23.0787

General Weather Info:
temprature is: 15.07 Degree Celcuis
Minum Temprature: 15.07 Degree Celcuis
Max Temprature: 15.07 Degree Celcuis
Feels like: 14.67 Degree Celcuis
Humidity : 78



DATA

City Info:
City name: Rietvlei
Country Code: ZA
Lon : 29.8739
Lat : -23.0787

General Weather Info:
temprature is: 15.07 Degree Celcuis
Minum Temprature: 15.07 Degree Celcuis
Max Temprature: 15.07 Degree Celcuis
Feels like: 14.67 Degree Celcuis
Humidity : 78
Pressure : 1027
Sea Level : 1027
Ground Level : 924
Sky : overcast clouds

PART-2 CONTAINERS

Question-2.1

1.1.

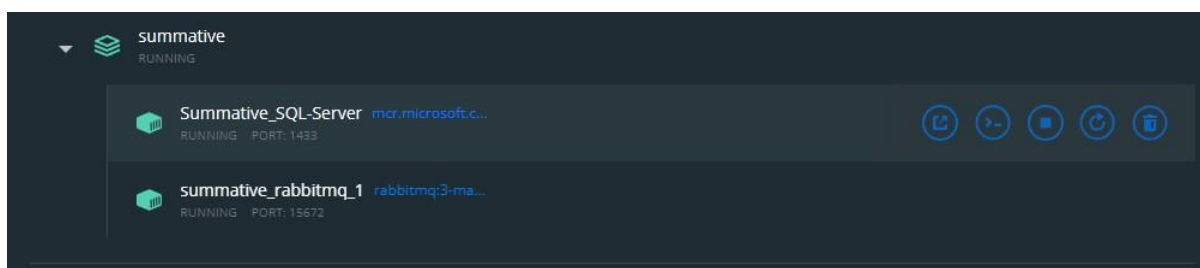
```
# Specify the compose format the file conforms to
version: "3"
# Specify the set of services your app is composed of
services:
# This is the db service
  db:
# The image to pull from docker hub
    image: mcr.microsoft.com/mssql/server:2019-latest
    hostname: Summative_SQL-Server
    container_name: Summative_SQL-Server
# Maps the HOST port of 1433 to the container port of 1433 (the
# default sql port)
    ports:
      - "1433:1433"
# Set environment variables
#   Set the ACCEPT_EULA variable to any value to confirm your
# acceptance of the
#   End-User Licensing Agreement.
#   Also set the password for the Sys Admin user.
    environment:
      - ACCEPT_EULA=Y
      - SA_PASSWORD=Password01
  rabbitmq:
# The image to pull from docker hub
    image: "rabbitmq:3-management"
# Maps ports HOST:Container, where 5672 is the port your apps will
# be connecting to rabbit
# 15672 is the port for the management portal enter localhost:15672
# in your browser
    ports:
      - "5672:5672"
      - "15672:15672"
# Create a volume for rabbit to persist its data to
    volumes:
      - "rabbitmq_data:/data"
#specifies the volumes to create as part of your app
volumes:
  rabbitmq_data:
```

```

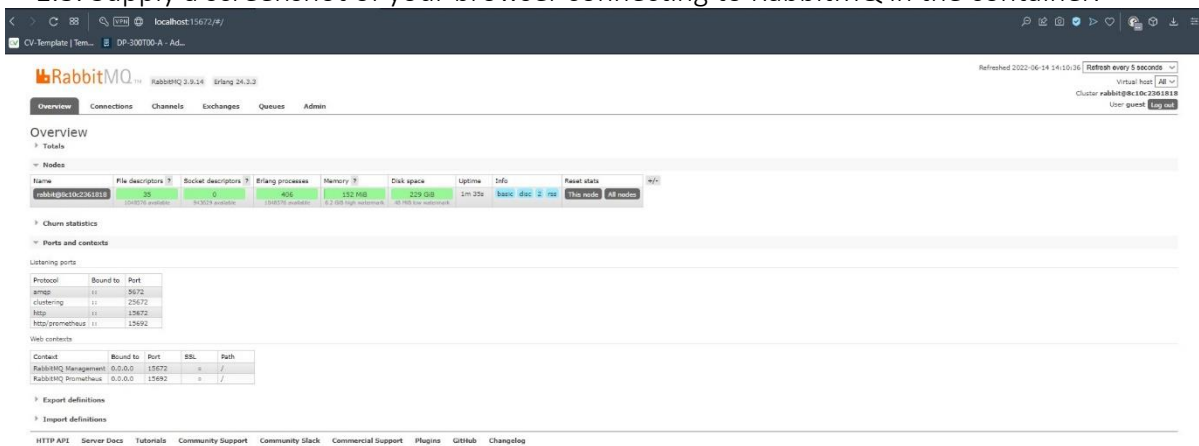
# Specify the compose format the file conforms to
version: "3"
# Specify the set of services your app is composed of
services:
# This is the db service
  db:
# The image to pull from docker hub
    image: mcr.microsoft.com/mssql/server:2019-latest
    hostname: Summative_SQL-Server
    container_name: Summative_SQL-Server
# Maps the HOST port of 1433 to the container port of 1433 (the
# default sql port)
    ports:
      - "1433:1433"
# Set environment variables
#   Set the ACCEPT_EULA variable to any value to confirm your
#   acceptance of the
#   End-User Licensing Agreement.
#   Also set the password for the Sys Admin user.
    environment:
      - ACCEPT_EULA=Y
      - SA_PASSWORD=Password01
  rabbitmq:
# The image to pull from docker hub
    image: "rabbitmq:3-management"
# Maps ports HOST:Container, where 5672 is the port your apps
# will be connecting to rabbit
# 15672 is the port for the management portal enter
# localhost:15672 in your browser
    ports:
      - "5672:5672"
      - "15672:15672"
# Create a volume for rabbit to persist its data to
    volumes:
      - "rabbitmq_data:/data"
#specifies the volumes to create as part of your app
volumes:
  rabbitmq_data:

```

1.2. Supply a screenshot of your containers running in the Docker Dashboard.



1.3. Supply a screenshot of your browser connecting to RabbitMQ in the container.



1.4. Supply a screenshot of SSMS connecting to the SQL Server in the container.



Question-2.2

2.1. Controller Action

```
using Microsoft.AspNetCore.Mvc;
using Products.DTO;
using Microsoft.Extensions.Hosting;
using Newtonsoft.Json;
using System;
using System.Threading;
using System.Threading.Tasks;
using System.Diagnostics;
using System.Net.Http;
using System.Collections.Generic;
using RabbitMQ.Client;
using System.Text;

namespace Products.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class productController : ControllerBase
    {
        [HttpGet]
        public IActionResult Get()
            => Ok("Product Service is running!");

        [HttpPost]
        public async Task<IActionResult> Order([FromBody] OrderDTO dto)
        {
            //check if the values being posted in postman is Null or empty price
            //and qty has to be checked to 0 because postman automatically posts empty
            //intergers or decimals to 0
            //Also checks if qty is less than 0
            if ((string.IsNullOrEmpty(dto.Item)) || (dto.Price.Equals(0)) ||
                (dto.Qty.Equals(0)) || (dto.Qty <= 0))
            {
                return Ok("Invalid Request");
            }

            Debug.WriteLine(dto.Item + dto.Price + dto.Qty);
            // Connect to RabbitMQ in your container
            var factory = new ConnectionFactory
            {
                Uri = new Uri("amqp://guest:guest@localhost:5672")
            };
            using var connection = factory.CreateConnection();
            using var channel = connection.CreateModel();

            // Connect to the "price-moved" queue on RabbitMQ
            channel.QueueDeclare(
                "Order-Placed",
```

```

        exclusive: false,
        autoDelete: false,
        arguments: null);


    // Serialize the MessageDto object into a json string
    var OrderEventAdded = JsonConvert.SerializeObject(dto);
    // Encode the json string into UTF8
    var body = Encoding.UTF8.GetBytes(OrderEventAdded);
    // Publish the message to the queue
    channel.BasicPublish("", "Order-Placed", null, body);

    return Ok("Order Placed");
}
}

```

2.2. The que was send inside the controller itself

2.3.

| Overview | | | | Messages | | | Message rates | | | +/- |
|--------------|---------|----------|--|----------|---------|-------|---------------|---------------|--------|-----|
| Name | Type | Features | State | Ready | Unacked | Total | incoming | deliver / get | ack | |
| Order-Placed | classic | |  idle | 0 | 0 | 0 | 0.00/s | 0.00/s | 0.00/s | |

2.5. Supply text copies of your startup.cs file in the Products Service.

```
namespace Products
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add
        services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllers();
        }

        // This method gets called by the runtime. Use this method to configure
        the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseHttpsRedirection();

            app.UseRouting();

            app.UseAuthorization();

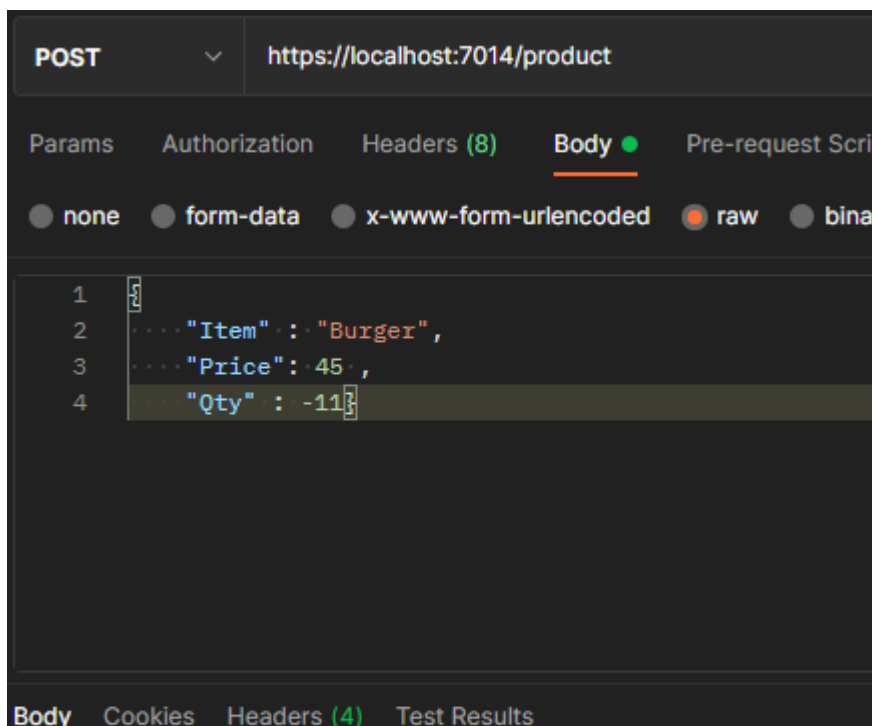
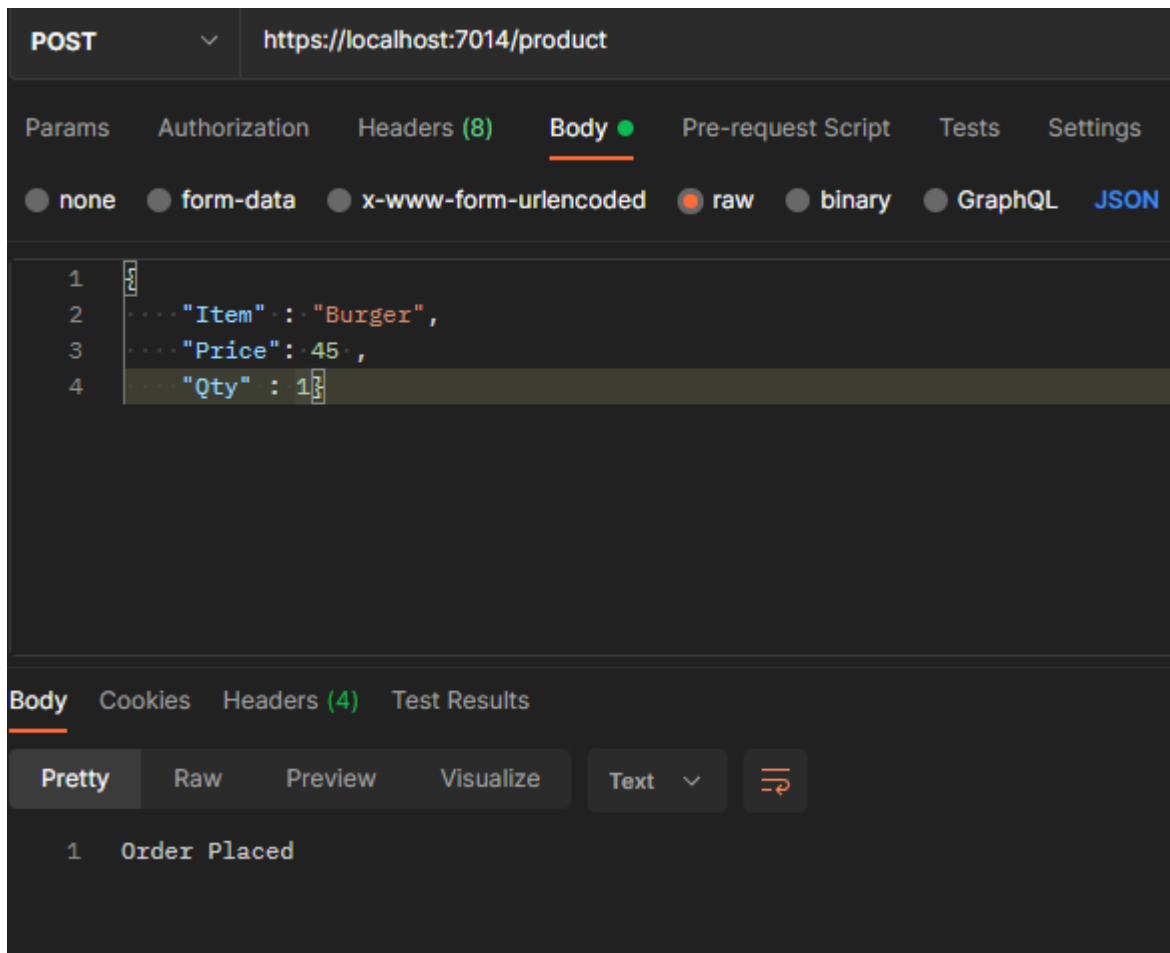
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
}
```

2.6. Supply text copies of your launchSettings.json file in the Products Service.

```
{
  "$schema": "https://json.schemastore.org/launchsettings.json",
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:4194",
      "sslPort": 44304
    }
  }
}
```

```
},
"profiles": {
  "Products": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "launchUrl": "product",
    "applicationUrl": "https://localhost:7014;http://localhost:5014",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
  "IIS Express": {
    "commandName": "IISExpress",
    "launchBrowser": true,
    "launchUrl": "product",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  }
}
}
```


2.7. Supply screenshots of successful and invalid requests and responses in the client (e.g. Postman) pertaining to the Products Service.



2.8. Supply text copies of any DTO's or other classes not already included in the Products Service.

```
namespace Products.DTO
{
    public class OrderDTO
    {
        public string Item { get; set; }
        public decimal Price { get; set; }
        public int Qty { get; set; }
    }
}
```

Question-2.3

3.1. Supply text copies of any Costing Service controller actions.

```
using Costing.Database;
using Costing.Database.Modals;
using Costing.DTO;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using static Costing.Helpers.HtmlList;

namespace Costing.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class CostingController : ControllerBase
    {
        private readonly InventoryDBContext _dbContext;
        private List<Message> balanceList;

        public CostingController(InventoryDBContext dbContext)
        {
            _dbContext = dbContext;
        }

        [HttpGet]
        public async Task<IActionResult> List()
        {
            List<CostRecords> transactionList = await
            _dbContext.Cost.ToListAsync();
            string htmlTable = Html.TransactionListHTML(transactionList,
            balanceList);

            return new ContentResult()
            {
                Content = htmlTable,
                ContentType = "text/html",
            };
        }
    }
}
```

3.2. Supply text copies of any Costing Service background services and any other services that you might have used to consume events/messages from the Ordered queue on RabbitMQ.

```
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Newtonsoft.Json;
using Costing.Database;
using RabbitMQ.Client;
using RabbitMQ.Client.Events;
using System;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using Costing.DTO;
using Costing.Database.Modals;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using System.Diagnostics;

namespace Costing.BackGroundService
{
    public class CostingBackgroundService : BackgroundService
    {
        private ConnectionFactory _connectionFactory;
        private IConnection _connection;
        private IModel _channel;
        private readonly IServiceScopeFactory _scopeFactory;

        private const string QueueName = "Order-Placed";

        public CostingBackgroundService(IServiceScopeFactory scopeFactory)
        {
            _scopeFactory = scopeFactory;
        }

        public override Task StartAsync(CancellationToken cancellationToken)
        {
            _connectionFactory = new ConnectionFactory
            {
                UserName = "guest",
                Password = "guest"
            };
            _connection = _connectionFactory.CreateConnection();
            _channel = _connection.CreateModel();
            _channel.QueueDeclare(QueueName,
                exclusive: false,
                autoDelete: false,
                arguments: null);
            _channel.BasicQos(0, 1, false);

            return base.StartAsync(cancellationToken);
        }
    }
}
```

```

    }

    protected override Task ExecuteAsync(CancellationToken stoppingToken)
    {
        stoppingToken.ThrowIfCancellationRequested();

        var timer = new Timer(CheckMessages, null, TimeSpan.Zero,
TimeSpan.FromSeconds(5));

        return Task.CompletedTask;
    }

    private async void CheckMessages(object state)
    {
        var consumer = new EventingBasicConsumer(_channel);

        consumer.Received += async (sender, evnt) =>
        {
            var body = evnt.Body.ToArray();
            var priceMovedEventData =
JsonConvert.DeserializeObject<Message>(Encoding.UTF8.GetString(body));
            using var scope = _scopeFactory.CreateScope();
            var dbContext =
scope.ServiceProvider.GetService<InventoryDBContext>();

            List<CostRecords> CostList = await dbContext.Cost.ToListAsync();

            CostRecords tRec = new CostRecords();
            var costprice = ((float)priceMovedEventData.Price) * 70 / 100;
            {
                tRec.Item = priceMovedEventData.Item;
                tRec.Price = ((float)priceMovedEventData.Price);
                tRec.Qty = priceMovedEventData.Qty;
                tRec.cost_price = costprice;
            };

            Debug.WriteLine(priceMovedEventData.Item);
            dbContext.Add(tRec);
            await dbContext.SaveChangesAsync();
        };

        _channel.BasicConsume(QueueName, true, consumer);
    }
}

```

3.3. **No Handler was necessary**

3.4. **No mapper Was necessary**

3.5. Supply text copies of your startup.cs file in the Costing Service.

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Costing.Database;
using Costing.BackGroundService;

namespace Costing {

public class Startup
{

    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to
the container.
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddControllers();
        services.AddDbContext<InventoryDbContext>();
        services.AddHostedService<CostingBackgroundService>();
    }

    // This method gets called by the runtime. Use this method to configure the
HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        app.UseHttpsRedirection();

        app.UseRouting();

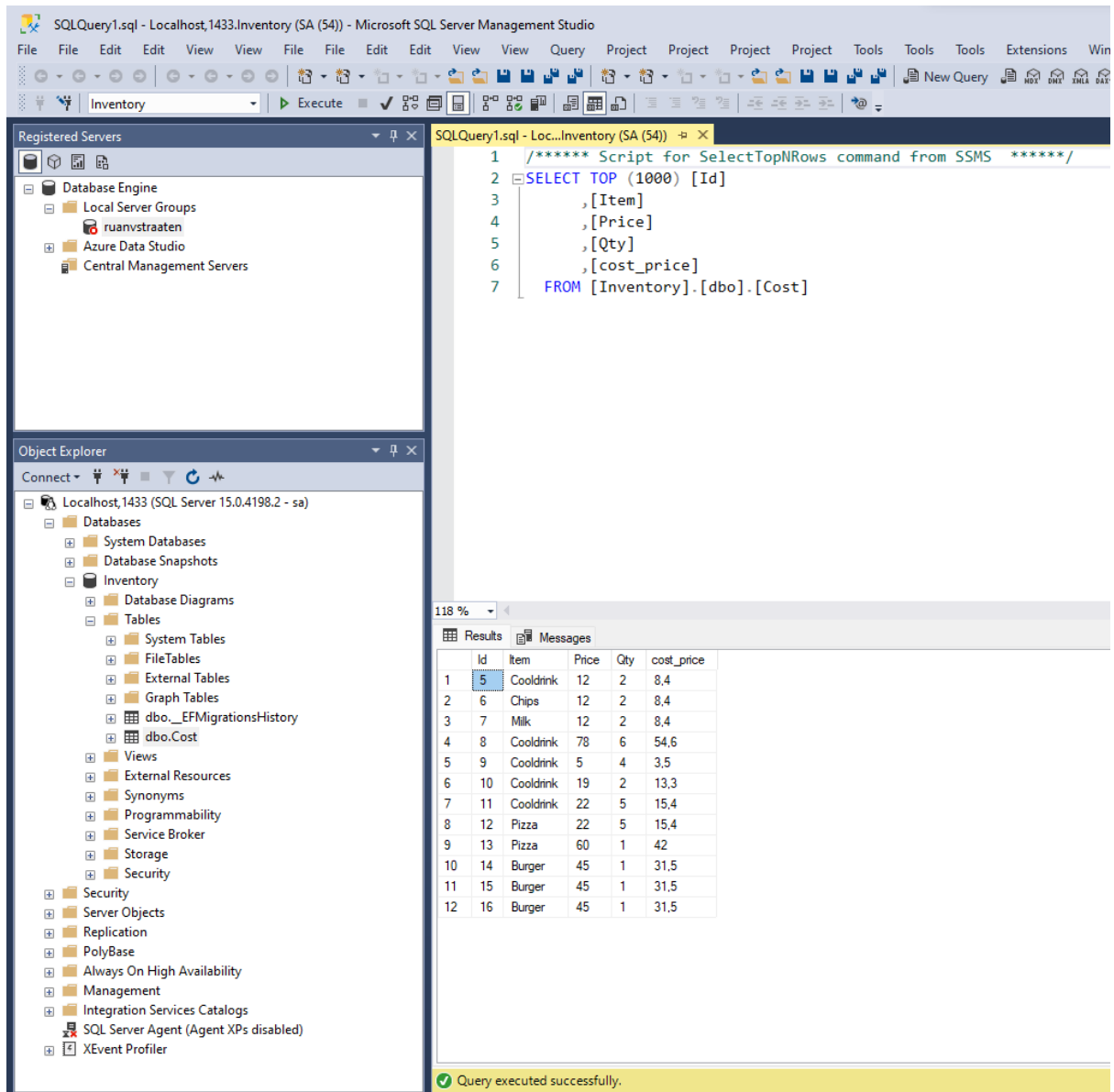
        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}
```

3.6. Supply text copies of your launchSettings.json file in the Costing Service.

```
{
  "$schema": "https://json.schemastore.org/launchsettings.json",
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:40861",
      "sslPort": 44361
    }
  },
  "profiles": {
    "Costing": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "launchUrl": "Costing",
      "applicationUrl": "https://localhost:7185;http://localhost:5185",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "launchUrl": "Costing",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

3.7. Supply screenshots of your Cost table's records displayed in SSMS (Total_Amount should not be a column in the table).



SQLQuery1.sql - Localhost,1433.Inventory (SA (54)) - Microsoft SQL Server Management Studio

File Edit View View View View View View Query Project Project Project Project Tools Tools Tools Extensions Win

Inventory Execute

Registered Servers

- Database Engine
 - Local Server Groups
 - ruanvstraaten
 - Azure Data Studio
 - Central Management Servers

Object Explorer

Connect

- Localhost,1433 (SQL Server 15.0.4198.2 - sa)
 - Databases
 - System Databases
 - Database Snapshots
 - Inventory
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.__EFMigrationsHistory
 - dbo.Cost
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Service Broker
 - Storage
 - Security
 - Security
 - Server Objects
 - Replication
 - PolyBase
 - Always On High Availability
 - Management
 - Integration Services Catalogs
 - SQL Server Agent (Agent XPs disabled)
 - XEvent Profiler

SQLQuery1.sql - Loc...Inventory (SA (54))

```
1 /***** Script for SelectTopNRows command from SSMS *****/
2 SELECT TOP (1000) [Id]
3     , [Item]
4     , [Price]
5     , [Qty]
6     , [cost_price]
7 FROM [Inventory].[dbo].[Cost]
```

118 %

Results Messages

| | Id | Item | Price | Qty | cost_price |
|----|----|-----------|-------|-----|------------|
| 1 | 5 | Cooldrink | 12 | 2 | 8,4 |
| 2 | 6 | Chips | 12 | 2 | 8,4 |
| 3 | 7 | Milk | 12 | 2 | 8,4 |
| 4 | 8 | Cooldrink | 78 | 6 | 54,6 |
| 5 | 9 | Cooldrink | 5 | 4 | 3,5 |
| 6 | 10 | Cooldrink | 19 | 2 | 13,3 |
| 7 | 11 | Cooldrink | 22 | 5 | 15,4 |
| 8 | 12 | Pizza | 22 | 5 | 15,4 |
| 9 | 13 | Pizza | 60 | 1 | 42 |
| 10 | 14 | Burger | 45 | 1 | 31,5 |
| 11 | 15 | Burger | 45 | 1 | 31,5 |
| 12 | 16 | Burger | 45 | 1 | 31,5 |

Query executed successfully.

3.8. Supply screenshots of the Costing service's HTML output (Total_Amount should be included in the HTML table).

The screenshot shows a web browser window titled 'Costing Service' with the address 'localhost:44361/Costing'. The page displays five tables of item data. Each table has columns: ID, Item, Price, Quantity, Costing Price, and Total.

| ID | Item | Price | Quantity | Costing Price | Total |
|----|--------|-------|----------|---------------|-------|
| 14 | Burger | 45 | 1 | 31,5 | 45 |
| 15 | Burger | 45 | 1 | 31,5 | 45 |
| 16 | Burger | 45 | 1 | 31,5 | 45 |

| ID | Item | Price | Quantity | Costing Price | Total |
|----|-------|-------|----------|---------------|-------|
| 6 | Chips | 12 | 2 | 8,4 | 24 |

| ID | Item | Price | Quantity | Costing Price | Total |
|----|-----------|-------|----------|---------------|-------|
| 5 | Cooldrink | 12 | 2 | 8,4 | 24 |
| 8 | Cooldrink | 78 | 6 | 54,6 | 468 |
| 9 | Cooldrink | 5 | 4 | 3,5 | 20 |
| 10 | Cooldrink | 19 | 2 | 13,3 | 38 |
| 11 | Cooldrink | 22 | 5 | 15,4 | 110 |

| ID | Item | Price | Quantity | Costing Price | Total |
|----|------|-------|----------|---------------|-------|
| 7 | Milk | 12 | 2 | 8,4 | 24 |

| ID | Item | Price | Quantity | Costing Price | Total |
|----|-------|-------|----------|---------------|-------|
| 12 | Pizza | 22 | 5 | 15,4 | 110 |
| 13 | Pizza | 60 | 1 | 42 | 60 |

3.9. Supply text copies of your model and context classes.

Cost Records

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;
```

```
namespace Costing.Database.Modals
```

```
{
    public class CostRecords
    {
        public int Id { get; set; }
        public string Item { get; set; }
        public float Price { get; set; }
        public int Qty { get; set; }
        public float cost_price { get; set; }
    }
}
```

DB Context

```
using Microsoft.EntityFrameworkCore;
using Costing.Database.Modals;
```

```
namespace Costing.Database
```

```
{
    public class InventoryDBContext : DbContext
    {
        public DbSet<CostRecords> Cost { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder options)
            => options.UseSqlServer(
                "Data Source=localhost,1433;" +
                "Persist Security Info=True;" +
                "User ID=sa;" +
                "Password=Password01;" +
                "Database=Inventory"
            );
    }
}
```

3.10. Supply text copies of any DTO's or other classes not already included.

DTO

```
namespace Costing.DTO
{
    public class Message
    {
        public string Item { get; set; }
        public decimal Price { get; set; }
        public int Qty { get; set; }
    }
}
```

HTML LIST

```
using Costing.Database.Modals;
using Costing.DTO;

namespace Costing.Helpers
{
    public class HtmlList
    {
        public static class Html
        {
            public static string previousSymbol = "";

            public static string TransactionListHTML(List<CostRecords> CostList,
                                                    List<Message> ItemList)
            {
                // Sort the transactions on Symbol
                List<CostRecords> SortedList = CostList.OrderBy(o =>
o.Item).ToList();

                string testHTML =
                "<html><head>" +
                "<title>Costing Service </title>" +
                "<style>table, th, td {border: 1px solid black;}</style></head>"
+
                "<body>";

                testHTML += buildTableHeading();

                // Table rows
                previousSymbol = "";
                foreach (CostRecords t in SortedList)
                {
                    // Test if symbol change and total should be printed - before
the current symbol's data is printed
                    if (previousSymbol != "" && t.Item != previousSymbol)
                    {
                        // Symbol changed. Print total. Then close off the table
and start a new table.

```

```

        testHTML += buildTableTotal();
        testHTML += buildTableHeading();
    }

    // Print current symbol's data
    testHTML += "<tr><td>" + t.Id + "</td><td>" + t.Item +
"</td><td>" + t.Price + "</td><td>" + t.Qty + "</td><td>" + t.cost_price +
"</td><td>" + (t.Price*t.Qty) + "</td></tr>";

    // Store the current symbol as previous symbol for the next
round
    previousSymbol = t.Item;
}

// After all the data printed, print the last currency's (which
is now stored in previousSymbol) total
testHTML += buildTableTotal();

testHTML += "</body></html>";

return testHTML;
}

internal static string TransactionListHTML(List<CostRecords>
transactionList, object balanceList)
{
    throw new NotImplementedException();
}

//th = table header
public static string buildTableHeading()
{
    return
("<table><tr><th>ID</th><th>Item</th><th>Price</th><th>Quantity</th><th>Costing
Price</th><th>Total</th></tr>");
}

public static string buildTableTotal()
{
    CostRecords t = new CostRecords();
    float balance = t.Price * t.Qty;

    return ("<tr><td>" + "</td><td>" + "</td><td>" + "</td><td>" +
"</td><td>" + "</tr>" +
        "</table><br>");
}
}
}
}

```

ANNEXURE-B

DECLARATION OF AUTHENTICITY

| | |
|--------------------|-----------|
| Module: | ADP631 |
| Assignment: | Summative |

I Ruan van straaten (FULL NAME) hereby declare that the contents of this assignment is entirely my own work with the exception of the following items:

(List the items and page numbers of work in this portfolio that are not your own work)

| Document/Activity/Section | Page Number |
|---------------------------|-------------|
| | |
| | |
| | |
| | |
| | |

I declare that I am familiar with and will abide to the Examination rules of CTU.


Signature

2022/06/20

Date