

# **Software Security Project**

---

## **Level 8 Cyber Resilience**

Unit title: Software Security (SCQF level 7)

Unit code: J4BH 34

# Assessment Objectives

## **Design, develop, test and document a secure solution to the problem**

Host the solution in a Google colab and in a Lambda function, accessed through an API endpoint.

1. **Design:** How solution to work.
2. **Develop:** Create solution, and a testing .
3. **Document:** Instructions on how to use solution.
4. **Secure the Lambda function:** Code is secure, meaning it's protected from unauthorized access and potential threats.
5. **Create, secure and test the API Endpoint:** A way for solution to be accessed through an API endpoint.

## THE PROBLEM

When people need to relocate to a new area, they may need to find schools within a reasonable distance of where they would live. This might determine their choice of location. If the process was automated, it would be quick to use and the information could come from one place, rather than a number of places.

Potentially, if a list could be stored, then it could be edited to reduce the list to a selection of schools that might be used in decision making about where to locate.

## USER STORIES

**AS** a user

**WHEN** I want to find the nearest schools to a given location

**THEN** I will be able to run a function in a Google Colab that will ask Google maps to send a list of nearby schools

**AND** I will be shown a list of schools in that area

**AND** this information will also be sent to a serverless function on AWS and saved in a schools file on S3

**AS** a user

**WHEN** I want to see all the schools in the file on AWS S3

**THEN** I will be able to run a function in a Google Colab that will send a request to get the full list,

**AND** I will see the list printed in my output

## ACCEPTANCE CRITERIA - USER STORIES

WHEN I send latitude 55.816555 and longitude -4.309890 and a radius of 4km

THEN the following list is added to the file:

Woodland Outdoor Kindergartens,

Hillpark Autism Unit,

Thornliebank Primary School,

Animal Man Parties,

Zippytotz

Hillpark Secondary School,

Eastwood Nursery School,

Smiley Stars Nursery - Mansewood, apple juice,

Source Wrestling School,

Orchard Park Nursery

WHEN I have entered a latitude of 87.543255 and longitude of 0, and a radius of 10km

THEN I will receive an empty list and nothing will be saved

WHEN I have entered a latitude of 99.999995 and longitude of 0, and a radius of 5km

THEN I will receive an error message telling me that the latitude is invalid (too big)

WHEN the request is sent to get the contents of the schools file on AWS S3

THEN a full list, **matching the file on S3**, is received

## Task 1a - Identify Main Threats and Vulnerabilities

Here are some potential threats and vulnerabilities that could affect the Google Maps functions in Colab, the Lambda function, and the API Gateway:

1. **Unauthorized access:** Hackers or unauthorized users may attempt to gain access to sensitive information stored in the solution, such as API keys or user data.
2. **Data interception:** Communication between the client and server may be intercepted, allowing attackers to view or modify data being transmitted.
3. **Denial of Service (DoS) attacks:** Attackers may flood the system with a high volume of requests, causing it to become unresponsive or crash.
4. **Injection attacks:** Malicious users may attempt to inject code or SQL queries into input fields, potentially leading to data breaches or system compromise.
5. **Insecure configurations:** Misconfigurations in the setup of the solution, such as weak encryption or improper access controls, could leave it vulnerable to attacks.
6. **Lack of input validation:** Failure to properly validate user input could open the system up to various vulnerabilities, including buffer overflows or code execution attacks.
7. **API abuse:** Attackers may attempt to abuse the API endpoints to perform unauthorized actions or extract sensitive information.
8. **Insider threats:** Malicious insiders or negligent employees may pose a threat to the security of the solution by intentionally or accidentally compromising sensitive data.

## Task 1b - Threat Table

Threat	Risk Level	Suggested Mitigation	Risk Level with Mitigation

## Task 1c - Risk Mitigation

To address each threat identified, the following actions can be taken:

1. Unauthorized access: Implement strong authentication mechanisms such as multi-factor authentication (MFA) and access controls based on least privilege principles. Regularly review and audit access logs to detect any unauthorized access attempts.
2. Data interception: Ensure all communication between the client and server is encrypted using HTTPS. Implement secure data transmission protocols such as SSL/TLS to prevent data interception.
3. Denial of Service attacks: Implement rate limiting and request validation mechanisms to mitigate the impact of DoS attacks. Use a content delivery network (CDN) to distribute traffic and improve scalability.
4. Injection attacks: Use parameterized queries and input validation to prevent injection attacks. Sanitize user input to remove any potentially malicious code.
5. Insecure configurations: Follow security best practices for configuring the solution, including strong encryption, proper access controls, and regular security audits.
6. Lack of input validation: Implement strict input validation and data sanitization routines to prevent input-based vulnerabilities. Use web application firewalls (WAF) to monitor and block potentially malicious traffic.
7. API abuse: Implement API key authentication and enforce rate limiting to prevent abuse of API endpoints. Regularly monitor API usage and review access logs for suspicious activity.
8. Insider threats: Implement role-based access controls (RBAC) to restrict access to sensitive data and functionalities based on job roles. Conduct regular employee training on security awareness and best practices to mitigate the risk of insider threats.

## Task 1d - Table Completion

Threat	Risk Level	Suggested Mitigation	Risk Level with Mitigation
Unauthorized access	High	Implement strong authentication and access controls, regularly audit access logs.	Medium
Data interception	High	Use HTTPS encryption for all communications, implement secure data transmission protocols.	Low
Denial of Service	Medium	Implement rate limiting and request validation, use a content delivery network (CDN) for scalability.	Low
Injection attacks	High	Use parameterized queries and input validation, sanitize user input to prevent code injection.	Medium
Insecure configurations	Medium	Follow security best practices for system configuration, conduct regular security audits.	Low
Lack of input validation	High	Implement strict input validation and data sanitization routines, use web application firewalls (WAF).	Medium
API abuse	High	Implement API key authentication, enforce rate limiting and request validation.	Medium
Insider threats	High	Implement role-based access controls, conduct employee training on security best practices.	Medium

## TASK 2 - Design, develop, test and document a software solution

### Task 2a - develop the program solution

Create a set of small functions that will work together. using Google Colab.

#### Colab/local functions:

##### Get Location Info:

- Write a function that asks the user for their location coordinates (latitude and longitude) and a radius.

##### Find Nearby Schools

- Use Google's API (a set of tools provided by Google) to find a list of schools near the location provided by the user.

The screenshot shows a Google Colab notebook interface. The title bar reads 'colab.research.google.com/drive/1fGxc0jNBoj54wq87atbwG3QRdyFLjyEC#scrollTo=6T9nNqKptJEb'. The notebook has a single code cell containing Python code. The code starts with installing 'googlemaps' and 'python-dotenv' via pip. It defines a function 'setAPI\_key()' to input an API key and store it in the environment variable 'API\_KEY'. The main function 'GetLocationInfo(latitude, longitude, radius)' takes three parameters. It checks if the inputs are floats and integers. It then creates a 'gmaps' object and initializes 'string\_schools' as an empty string. It iterates over 'places\_result['results']' and appends each school's information (latitude, longitude, radius, name) to 'string\_schools'. If an exception occurs, it prints the error message. Finally, it prints 'test1:' and 'test2:' followed by the results of calling 'GetLocationInfo' with specific coordinates and radii. The output pane shows the resulting list of schools for both test cases.

```
!pip install googlemaps
!pip install python-dotenv
from dotenv import load_dotenv
import os
from google.colab import output
import googlemaps

def setAPI_key():
    key = input("Enter the API key: ")
    os.environ['API_KEY'] = key
    output.clear()

def GetLocationInfo(latitude, longitude, radius):
    gmaps = googlemaps.Client(os.environ.get('API_KEY'))
    string_schools = ''
    if not isinstance(latitude,(float)) and not isinstance(latitude,(int)):
        raise TypeError('Invalid latitude data type')
    if not isinstance(longitude,(float)) and not isinstance(longitude,(int)):
        raise TypeError('Invalid longitude data type')
    if not isinstance(radius,(float)) and not isinstance(radius,(int)):
        raise TypeError('Invalid radius data type')
    if latitude > 90.0 or latitude < -90.0:
        raise ValueError("Invalid latitude value.")
    if longitude > 180.0 or longitude < -180.0:
        raise ValueError("Invalid longitude value.")
    if radius <= 0:
        raise ValueError('Check the value of radius; is radius a non-negative?')
    try:
        places_result = gmaps.places_nearby(location=str(latitude) + ',' + str(longitude), radius = radius*1000, type = "school")
        for place in places_result['results']:
            string_schools = string_schools + str(latitude) + ', ' + str(longitude) + ', ' + str(radius) + ', ' + str(place['name']) + '\n'
    except Exception as e:
        print("Error: " + str(e))
    return string_schools

setAPI_key()
print('test1:')
result1 = GetLocationInfo(55.816555,-4.309890,4)
print(result1)
print('test2:')
result2 = GetLocationInfo(87.543255,0,10)
print(result2)

test1:
55.816555, -4.30989, 4, Hutchesons' Grammar School
55.816555, -4.30989, 4, Eastwood High School
55.816555, -4.30989, 4, Glasgow Club Bellahouston
55.816555, -4.30989, 4, Bellahouston Academy
55.816555, -4.30989, 4, Pro-Soccer
55.816555, -4.30989, 4, St Vincent's Primary School
55.816555, -4.30989, 4, Cleeves Primary School
55.816555, -4.30989, 4, Thornliebank Primary School
55.816555, -4.30989, 4, Our Lady of the Mission Primary School
55.816555, -4.30989, 4, Hillpark Autism Unit
55.816555, -4.30989, 4, Fotheringay Centre
55.816555, -4.30989, 4, Animal Man Parties
55.816555, -4.30989, 4, Woodland Outdoor Kindergartens
55.816555, -4.30989, 4, Kumon Maths & English
55.816555, -4.30989, 4, Carmichael Nursery School
55.816555, -4.30989, 4, Wendy King
55.816555, -4.30989, 4, Burnbrae Children's Centre
55.816555, -4.30989, 4, Zippytots
55.816555, -4.30989, 4, Tinto Primary School
55.816555, -4.30989, 4, St Vincent's Autism Unit Co St Vincent's Primary

test2:
```

```

SoftwareSecurityProject.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Test
Q ipg install geogmaps
ipg install python-dotenv
from dotenv import load_dotenv
import os
from geogmaps import output
import geogmaps

def setAPI_key():
    key = input("Enter the API key: ")
    os.environ['API_KEY'] = key
    output.clear()

def GetLocationInfo(latitude, longitude, radius):
    gmaps = geogmaps.Client(os.environ.get('API_KEY'))
    string_shells = ''
    if not isinstance(latitude,(float)) and not isinstance(latitude,(int)):
        raise TypeError('Invalid latitude data type')
    if not isinstance(longitude,(float)) and not isinstance(longitude,(int)):
        raise TypeError('Invalid longitude data type')
    if not isinstance(radius,(float)) and not isinstance(radius,(int)):
        raise TypeError('Invalid radius data type')
    if latitude > 90.0 or latitude < -90.0:
        raise ValueError('Invalid latitude value')
    if longitude > 180.0 or longitude < -180.0:
        raise ValueError('Invalid longitude value')
    if radius <= 0:
        raise ValueError('Check the value of radius; is radius a non-negative?')
    try:
        places_result = gmaps.places_nearby(vicinity=f'{latitude} {longitude}', radius = radius*1000, type = "school")
        for place in places_result['results']:
            string_shells += str(latitude) + ', ' + str(longitude) + ', ' + str(radius) + ', ' + str(place['name']) + '\n'
        return string_shells
    except Exception as e:
        print('Error: ' + str(e))
    return string

setAPI_key()

print('test1:')
result1 = GetLocationInfo(55.816555, -4.309888,4)
print(result1)

print('test2:')
result2 = GetLocationInfo(56.816555, -3.309888,7)
print(result2)

print('test3:')
result3 = GetLocationInfo(58.816555, -3.309888,9)
print(result3)

#print('test4:')
#result4 = GetLocationInfo(87.543255,8,10)
#print(result4)

#print('test5:')
#result5 = GetLocationInfo(88.000005,8,5)
#print(result5)

text1:
55.816555, -4.309889, 4, Hatchescant's Grammar School
55.816555, -4.309889, 4, Eastwood High School
55.816555, -4.309889, 4, Glaziers Club Ballilaouston
55.816555, -4.309889, 4, St Columba's Academy
55.816555, -4.309889, 4, Pra-Soccer
55.816555, -4.309889, 4, St Vincent's Primary School
55.816555, -4.309889, 4, Cleaves Primary School
55.816555, -4.309889, 4, Thornliebank Primary School
55.816555, -4.309889, 4, Hillspark Autism Unit
55.816555, -4.309889, 4, Fatherjagay Centre
55.816555, -4.309889, 4, Animal Man Parties
55.816555, -4.309889, 4, Headland Outdoor Kindergarten
55.816555, -4.309889, 4, Kameo Marim & Gringlin
55.816555, -4.309889, 4, Merricott Nursery School
55.816555, -4.309889, 4, Wendy King
55.816555, -4.309889, 4, Burnbrae Children's Centre
55.816555, -4.309889, 4, Ziggycats
55.816555, -4.309889, 4, Tista Primary School
55.816555, -4.309889, 4, St Vincent's Autism Unit Co St Vincent's Primary

text2:
58.816555, -3.309889, 7, Paghamerry C Of E Primary School
58.816555, -3.309889, 7, Feniton Primary School
58.816555, -3.309889, 7, Powerbeat & RIB
58.816555, -3.309889, 7, Play & Go Primary School
58.816555, -3.309889, 7, Plymtree Primary School
58.816555, -3.309889, 7, Bradenbury Church of England Primary School
58.816555, -3.309889, 7, The Forest School at Excot
58.816555, -3.309889, 7, Auliccross Primary School
58.816555, -3.309889, 7, Kestrelshare C Of E Primary School
58.816555, -3.309889, 7, Hanny Bears Nursey and Pre-school
58.816555, -3.309889, 7, Whimple Preschool
58.816555, -3.309889, 7, Whimple Primary School
58.816555, -3.309889, 7, A & C Day Nursey

text3:
58.816555, -3.309889, 9, North Walls Primary School
58.816555, -3.309889, 9, North Walls Junior High School
58.816555, -3.309889, 9, Little Wards

```

## Send Data to AWS Lambda:

- Sends the list of nearby schools to a function you've made on AWS Lambda. AWS Lambda is a service provided by Amazon for running code without managing servers.

```

import requests
import json

def save_a_file(result):

    if not isinstance(result,(str)):
        raise TypeError('Invalid file data type')

    url = "https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda"

    headers = {
        "Content-Type": "application/json",
        "x-api-key": input("Enter the API key so that it isn't stored in the notebook: ")
    }

    body = {
        "data": {
            "action": "store",
            "schools": result
        }
    }

    response = requests.post(url, headers=headers, data=json.dumps(body))
    return response

result1 = GetLocationInfo(55.816555,-4.309890,4)
save_a_file(result1)

```

Enter the API key so that it isn't stored in the notebook: lVjE3I0uUdaU72VgUhws45je0sEeEUurhpWKTs19  
<Response [200]>

```

import requests
import json

def get_a_file():

    url = "https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda"

    headers = {
        "Content-Type": "application/json",
        "x-api-key": input("Enter the API key so that it isn't stored in the notebook: ")
    }

    body = {
        "data": {
            "action": "get"
        }
    }

    response = requests.post(url, headers=headers, data=json.dumps(body))
    return response.text

get_a_file()

```

Enter the API key so that it isn't stored in the notebook: lVjE3I0uUdaU72VgUhws45je0sEeEUurhpWKTs19  
'55.816555, -4.30989, 4, Hutchesons' Grammar School\n55.816555, -4.30989, 4, Eastwood High School\n55.816555, -4.30989, 4, Glasgow Club Bellahouston\n55.816555, -4.30989, 4, Bellahouston Academy\n55.816555, -4.30989, 4, Pro-Soccer\n55.816555, -4.30989, 4, St Vincent's Primary School\n55.816555, -4.30989, 4, Clevees Primary School\n55.816555, -4.30989, 4, Thornliebank Primary School\n55.816555, -4.30989, 4, Our Lady of the Mission Primary School\n55.816555, -4.30989, 4, Hillpark Autism Unit\n55.816555, -4.30989, 4, Fotheringay Centre\n55.816555, -4.30989, 4, Animal Man Parties\n55.816555, -4.30989, 4, Woodland Outdoor Kindergartens\n55.816555, -4.30989, 4, Kumon Maths & English\n55.816555, -4.30989, 4, Carmichael Nursery School\n55.816555, -4.30989, 4, Wendy King\n55.816555, -4.30989, 4, Burnbrae Children's Centre\n55.816555, -4.30989, 4, Zippytotz\n55.816555, -4.30989, 4, Tinto Primary School\n55.816555, -4.30989, 4, St Vincent's Autism Unit Co St Vincent's Primary\n55.816555, -4.30989, ...'

## AWS Lambda Function:

### Receive a request to store a new list of schools or get the whole list

- make sure you have a function set up on AWS Lambda to receive the request to either store a new set of schools or to get the whole list from the file and return it

### Format the list of schools ready to add to a CSV (Comma-Separated Values) text file.

- this means putting each school's information in a structured format separated by commas.

### Read the file contents and return the list

- able to connect to the file
- read all the contents of the file
- delete a name of a school from the file on S3

Copy of Software SecurityProject.ipynb

```
# functions.py
import boto3
import io
import os

def get_S3_client():
    s3_resource = boto3.client('s3')
    return s3_resource

def get_file(s3_client, filename):
    # get the file from the bucket
    if filename[-4:] != '.csv':
        raise ValueError('Check the value of filename')
    try:
        file_object = s3_client.get_object(Bucket=os.environ.get('BUCKET_NAME'), Key=filename)
        data_file = file_object['Body'].read()
    except Exception as e:
        print("Error: " + str(e))
    return str(data_file.decode("utf-8"))

def save_a_file(filedata, filename):
    # create a new file by uploading the data with the new filename
    s3_client = get_S3_client()
    if filename[-4:] != '.csv':
        raise ValueError('Check the value of filename')
    if len(get_file(s3_client, filename)) > 0:
        filedata = get_file(s3_client, filename) + filedata
    response = s3_client.put_object(Bucket=os.environ.get('BUCKET_NAME'), Body=filedata, Key=filename)

def delete_a_school(filedata, name):
    if not isinstance(filedata, str):
        raise TypeError('Invalid filedata data type')
    if not isinstance(name, str):
        raise TypeError('Invalid name data type')
    if len(filedata) < len(name):
        raise ValueError('Check the value of filedata')
    data = filedata.split('\n')
    for x in data:
        if x.find(name) != -1:
            data.remove(x)
    return '\n'.join(data)
```

Copy of Software SecurityProject.ipynb

```
# lambda_function.py
# handle the request
import boto3
import io
import os
import json
# from functions import get_file, get_S3_client, save_a_file, delete_a_school

def lambda_handler(event, context):
    s3_client = get_S3_client()
    if "body" in event.keys():
        request = event["body"]
        if type(request) is not dict:
            request = json.loads(request)
        # check that there was some data in the body, get the values and run the function to get the result
        if request is not None and "data" in request.keys():
            # get the data from the data object
            try:
                data = request["data"]
                action = data["action"]
            except KeyError:
                print("KeyError")
                return "KeyError"
            # now the data has been collected, it can run the functions and set the statuscode to 200 (success)
            if action == "get":
                return_data = get_file(s3_client, 'schools.csv')
                statuscode = 200
            elif action == "post":
                for place in data["results"]:
                    string_shools = string_shools + str(latitude) + ',' + str(longitude) + ',' + str(radius) + ',' + str(place['name']) + '\n'
                return_data = save_a_file(string_shools, 'schools.csv')
                statuscode = 200
            elif action == "delete":
                try:
                    school = data["school"]
                    data = get_file(s3_client, 'schools.csv')
                    new_data = delete_a_school(data, school)
                    save_a_file(new_data, 'schools.csv')
                    return_data = "School deleted"
                except KeyError:
                    print("KeyError")
                    return "KeyError"
                statuscode = 200
            else:
                return_data = "Unsupported operation"
                statuscode = 400
            else:
                return_data = "Unable to get data"
                statuscode = 404
        # now it can return the result in a JSON object with some security settings in the headers
        return {
            'statusCode': statuscode,
            'headers': {
                'Content-Type': 'application/json',
                'Access-Control-Allow-Headers': 'Content-Type,X-API-Key',
                'Access-Control-Allow-Methods': 'POST',
                'Access-Control-Allow-Origin': '*'
            },
            'body': return_data
        }
```

## TASK 2b - Create a serverless function on AWS Lambda, with an API gateway, and secure it

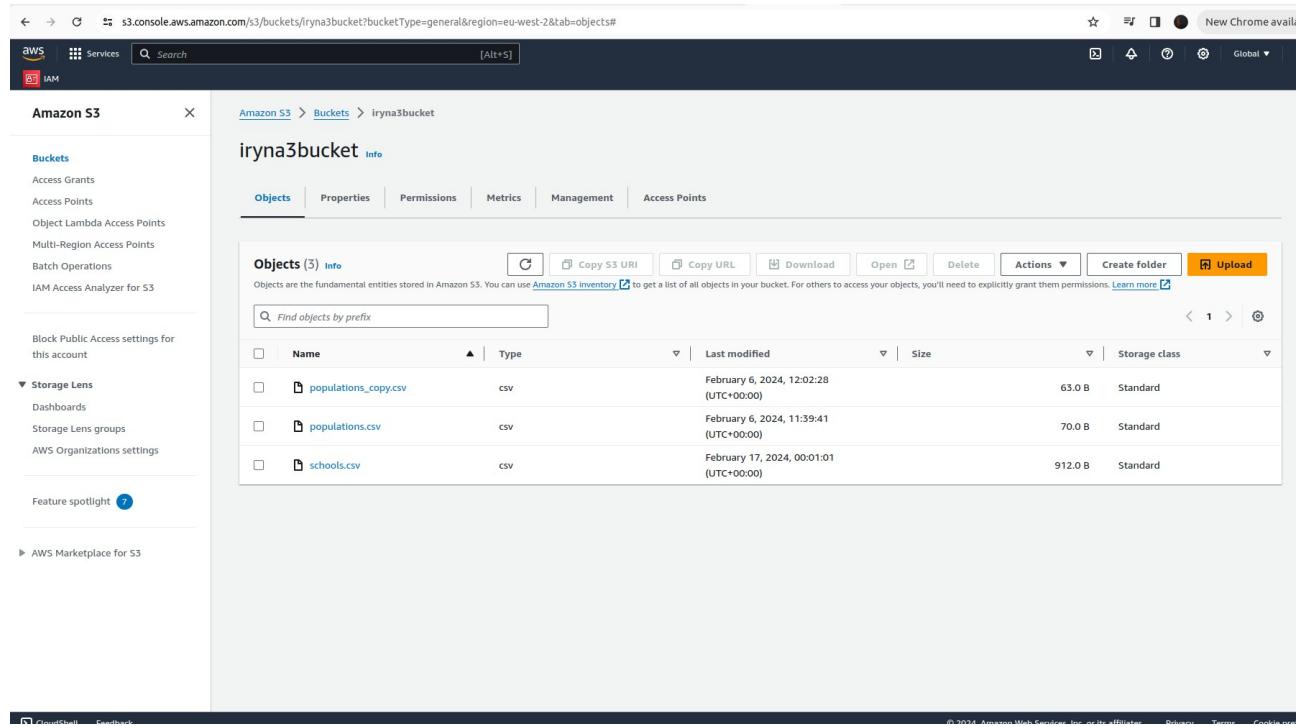
### 1. Create Serverless Function:

- Use AWS Lambda, a service provided by Amazon, to build a function that doesn't need a server to run.

The screenshot shows the AWS Lambda Function Overview page for the function 'FindNearestSchoolsLambda'. The top navigation bar includes 'Services' and 'Search [Alt + S]'. A green success message box states 'Successfully updated the function FindNearestSchoolsLambda.' Below the message, the function name 'FindNearestSchoolsLambda' is displayed with a 'Throttle' button, 'Copy ARN' button, and 'Actions' dropdown. A note indicates that the trigger 'FindNearestSchoolsLambda-API' was successfully added. The 'Function overview' section shows a diagram where 'FindNearestSchoolsLambda' is connected to 'API Gateway'. Buttons for '+ Add destination' and '+ Add trigger' are visible. On the right, there are fields for 'Description', 'Last modified' (4 hours ago), 'Function ARN' (arn:aws:lambda:eu-west-2:333267215096:function:FindNearestSchoolsLambda), and 'Function URL' (Info). Below the overview, tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions' are present. The 'Code' tab is active, showing the 'Code source' section with an 'Info' button and a file browser interface. The code editor displays the file 'lambda\_function.py' with Python code. The sidebar shows the environment variables and the file structure. The bottom sections include 'Code properties' (Info) with package size (1.8 kB), SHA256 hash (S3z0R88pKM5HhfUIA3c7vT+qDhWPhmsKBjQsgQaMM4=), and last modified (February 29, 2024 at 10:43 AM GMT); 'Runtime settings' (Info) with runtime (Python 3.12), handler (lambda\_function.lambda\_handler), and architecture (x86\_64); and a 'Runtime management configuration' link.

## 2. Access S3:

- Make sure your Lambda function can read and write files in Amazon S3, a storage service provided by Amazon.



The screenshot shows the AWS S3 console interface. The left sidebar has a 'Buckets' section with links like 'Access Grants', 'Access Points', 'Object Lambda Access Points', etc. The main content area shows the 'iryna3bucket' details. The 'Objects' tab is selected, displaying three CSV files: 'populations\_copy.csv', 'populations.csv', and 'schools.csv'. The table includes columns for Name, Type, Last modified, Size, and Storage class.

Name	Type	Last modified	Size	Storage class
populations_copy.csv	csv	February 6, 2024, 12:02:28 (UTC+00:00)	63.0 B	Standard
populations.csv	csv	February 6, 2024, 11:39:41 (UTC+00:00)	70.0 B	Standard
schools.csv	csv	February 17, 2024, 00:01:01 (UTC+00:00)	912.0 B	Standard

Screenshot of the AWS Lambda console showing the function "FindNearestSchoolsLambda".

**Function Overview:**

- Successfully updated the function FindNearestSchoolsLambda.
- Layers: 0
- API Gateway: + Add destination
- + Add trigger
- Last modified: 3 hours ago
- Function ARN: arn:aws:lambda:eu-west-2:333267215090:function:FindNearestSchoolsLambda
- Function URL: Info

**Test Tab:**

Executing function: succeeded (logs)

The area below shows the last 4 KB of the execution log.

```

"Access-Control-Allow-Methods": "POST",
"Access-Control-Allow-Origin": **",
},
"body": "+55.816555, -4.39989, 4, Hutchesons' Grammar School\n+55.816555, -4.39989, 4, Eastwood High School\n+55.816555, -4.39989, 4, Glasgow Club Bellahouston\n+55.816555, -4.39989, 4, Bellahouston Primary School\n+55.816555, -4.39989, 4, St. Vincent's Primary School\n+55.816555, -4.39989, 4, Clevedon Primary School\n+55.816555, -4.39989, 4, Thornbank Primary School\n+55.816555, -4.39989, 4, Our Lady of the Assumption Primary School\n+55.816555, -4.39989, 4, Hillpark Autism Unit\n+55.816555, -4.39989, 4, Fotheringay Centre\n+55.816555, -4.39989, 4, Animal Man Parciale\n+55.816555, -4.39989, 4, Woodland Outdoor Kindergarten\n+55.816555, -4.39989, 4, Kumei Maths & English\n+55.816555, -4.39989, 4, Carmichael Nursery School\n+55.816555, -4.39989, 4, Wesley King\n+55.816555, -4.39989, 4, Burnbrae Children's Centre\n+55.816555, -4.39989, 4, Zippytots\n+55.816555, -4.39989, 4, Tinto Primary School\n+55.816555, -4.39989, 4, St Vincent's Autism Unit Co St Vincent's Primary\n"
}

```

**Summary:**

Code SHA-256: 53zoR8RpKM5HfuiA3c7vT+qDdHWPhmsKBjQsgQaMM4=	Execution time: 1 hour ago [February 29, 2024 at 03:20 PM GMT]
Request ID: 20fb083a-a3d5-41f8-a3f9-c00db393297f	Function version: \$LATEST
Init duration: 274.21 ms	Duration: 2752.68 ms
Billed duration: 2753 ms	Resources configured: 128 MB
Max memory used: 81 MB	

**Log output:**

The section below shows the logging calls in your code. [Click here](#) to view the corresponding CloudWatch log group.

```

START RequestId: 20fb083a-a3d5-41f8-a3f9-c00db393297f Version: $LATEST
END RequestId: 20fb083a-a3d5-41f8-a3f9-c00db393297f
REPORT RequestId: 20fb083a-a3d5-41f8-a3f9-c00db393297f Duration: 2752.68 ms Billed Duration: 2753 ms Memory Size: 128 MB Max Memory Used: 81 MB Init Duration: 274.21 ms

```

**Test event:**

To Invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

Test event action:  Create new event  Edit saved event

Event name: test-get

Event JSON:

```

1 v
2 v
3 v
4 v
5 v
6 v
7 v

```

Format JSON

### 3. Trigger with API Gateway:

- Set up your Lambda function to be triggered (or activated) by an API Gateway. This means it will be called when someone makes a request through an API.

The screenshot shows the AWS Lambda console interface. At the top, a green banner indicates that the function has been successfully updated. Below this, the function name 'FindNearestSchoolsLambda' is displayed, along with navigation links for Lambda > Functions > FindNearestSchoolsLambda. On the right, there are buttons for Throttle, Copy ARN, and Actions.

**Function overview**

**Description:** - Last modified 3 hours ago

**Function ARN:** arn:aws:lambda:eu-west-2:533207215096:function:FindNearestSchoolsLambda

**Function URL:** -

**Triggers (1)**

- Trigger:** API Gateway: FindNearestSchoolsLambda-API
  - API endpoint: <https://qvk98549.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda>
  - Details:
    - API key: lvjE3l0uUdaU72VgUHws45jeOsEEUurhpWKTs9
    - API type: REST
    - Authorization: NONE
    - Method: ANY
    - Resource path: /FindNearestSchoolsLambda
    - Service principal: apigateway.amazonaws.com
    - Stage: default
    - Statement ID: lambda-41c512ec-11d0-46ac-a6af-1dc888a6269f

**Configuration**

**General configuration:**

- Triggers (selected)
- Permissions
- Destinations
- Function URL
- Environment variables
- Tags
- VPC
- Monitoring and operations tools
- Concurrency
- Asynchronous Invocation
- Code signing
- RDS databases
- File systems
- State machines

#### 4. Secure with API Key:

- Make sure your API Gateway is protected by an API key. This means people need a special key to access your API and trigger your Lambda function.

The screenshot shows the AWS API Gateway API Keys page. A specific API key named 'readFilesLambdaFunction-Key' is selected. The key details include:
 

- ID: 42fB846
- Description: Created by AWS Lambda
- Creation date: February 07, 2024, 14:10 (UTC+00:00)
- Status: Active
- Associated APIs: readFilesLambdaFunction-API
- Associated stages: default

#### 5. Function's Tasks:

- Your Lambda function should be able to do two things:
  - Accept a request to store data in a specific format.
  - Accept a request to retrieve all the data from a file.

The screenshot shows the AWS Lambda console interface. At the top, there's a search bar and a [Alt+S] keybinding indicator. The main area displays the 'FindNearestSchools' function, which is associated with an 'API Gateway' trigger. Below the function name are 'Layers' and '(0)' indicating no layers are currently attached. To the right, there are details about the function's last modification (4 hours ago) and its ARN (arn:aws:lambda:eu-west-2:533267215096:function:FindNearestSchools). A 'Function URL' link is also present. The bottom navigation bar includes tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The 'Code' tab is selected, showing the code source in a code editor window. The code is written in Python and defines a lambda function that handles various actions like getting files, saving schools, and deleting schools based on the input parameters.

```
# handle the request
def lambda_handler(event, context):
    s3 = boto3.client('s3')
    # Import the required module
    import json
    # Get the file from S3
    file_name = event['File']
    # Save the file to a local variable
    s3.download_file(file_name)
    # Load the file into memory
    file_content = open(file_name, 'r').read()
    # Convert the file content into a JSON object
    file_content = json.loads(file_content)
    # Create a list of schools
    schools = []
    # Loop through each school in the file
    for school in file_content:
        # Add the school to the list
        schools.append(school)
    # Save the schools to a CSV file
    with open('schools.csv', 'w') as f:
        writer = csv.writer(f)
        writer.writerow(['Name', 'Address'])
        for school in schools:
            writer.writerow([school['Name'], school['Address']])
    # Delete the file from S3
    s3.delete_object(file_name)
    # Return the file
    return file_content
```

The screenshot shows the AWS Lambda console interface. At the top, there's a search bar and a [Alt+S] keyboard shortcut. Below the header, the function name 'FindNearestSchools' is displayed, along with 'Throttle', 'Copy ARN', and 'Actions' buttons. A 'Function overview' section includes a 'Diagram' tab (selected), a 'Template' tab, and a box showing the function's execution environment. The diagram shows the function connected to an 'API Gateway' and a 'Layers' section with '(0)'. There are '+ Add destination' and '+ Add trigger' buttons. To the right, a 'Description' panel shows the last modified time as '4 hours ago' and provides the Function ARN and URL. Below the main content area are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code source' tab is selected, showing the code editor with the 'functions.py' file open. The code is as follows:

```
1  #!/usr/bin/python
2  import boto3
3  import io
4  import os
5
6  def get_s3_client():
7      s3_resource = boto3.client('s3')
8      return s3_resource
9
10 def get_file(s3_client, filename):
11     file = s3_client.get_object(Bucket=os.environ.get('BUCKET NAME'), Key=filename)
12     data_file = s3_client.get_object(Bucket=os.environ.get('BUCKET NAME'), Key=filename).read()
13     return str(data_file.decode("utf-8"))
14
15 def save_a_file(filedata, filename):
16     s3_client = get_s3_client()
17     response = s3_client.put_object(Bucket=os.environ.get('BUCKET NAME'), Body=filedata, Key=filename)
18
19 def delete_a_school(filedata,filename):
20     if not isinstance(filedata,(str)):
21         raise TypeError('Invalid filedata data type')
22     if not isinstance(filename,(str)):
23         raise TypeError('Invalid filename data type')
24     if len(filedata) < len(filename):
25         raise ValueError('Check the value of filedata')
26     data = filedata.split('\n')
27     for x in data:
28         if x.find(filename) != -1:
29             data.remove(x)
30     return '\n'.join(data)
```

## **TASK 3 - Test software to identify security issues using appropriate methodologies AND THOUGHOUT THE ENTIRE PROJECT**

### **1. Test for Security Issues:**

- Check the software to find any security problems. This means looking for ways that unauthorised people could access or mess with the system.

**Test 1.** For increased security, I configured multi-factor authentication (MFA) to help protect AWS resources for the AWS account root user and IAM user.

MFA added extra security because it requires users to provide unique authentication from an AWS supported MFA mechanism in addition to their regular sign-in credentials when they access AWS websites or services.

I used virtual authenticator application Authy Desktop that runs on my PC device and emulates a physical device. Virtual authenticator apps implement the time-based one-time password (TOTP) algorithm and support multiple tokens on a single device. The user must type a valid code from the device on a second webpage during sign-in. Each virtual MFA device assigned to a user must be unique. A user can't type a code from another user's virtual MFA device to authenticate.

The screenshot shows the AWS sign-in interface. On the left, the 'Sign in as IAM user' form is displayed with fields for Account ID (533267215096), IAM user name (root), and Password. There is also a 'Remember this account' checkbox and a 'Sign in' button. Below the form are links for 'Sign in using root user email' and 'Forgot password?'. On the right, there is a dark sidebar advertisement for Amazon Redshift. It features the text 'ANALYTICS Start on Amazon Redshift for Free' and 'Try a two month free trial of our DC2.Large node with 750 free hours per month'. It includes a 'Learn more >' link and a small icon depicting a database cylinder, a cloud, and a gauge.

The screenshot shows two windows side-by-side. On the left is the AWS Multi-factor Authentication page, which asks for an MFA code. A user has entered '272891' into the input field. Below the input field is a blue 'Submit' button. To the right of the input field is a 'Cancel' link. At the bottom of the page are links for 'Terms of Use' and 'Privacy Policy'. On the right is the Twilio Authy desktop application window. It displays the AWS logo and the number '272 891' in large digits. Below the number, it says 'Changes in 19 Seconds'. At the bottom of the window are three buttons: 'Tokens' (red), 'Requests' (green), and 'Settings' (blue).

When unauthorised people input wrong login, password or the time-based one-time password (TOTP) they couldn't access AWS system.

The screenshot shows the AWS sign-in page. A red box highlights the 'Authentication failed' message: 'Your authentication information is incorrect. Please try again.' Below this message is the 'Root user sign in' section. It includes fields for 'Email' (idunets@yahoo.com), 'Password', and a 'Sign in' button. To the right of the sign-in form is a dark sidebar for 'Amazon EC2 Inf2 Instances' featuring a 'Sign up now' button and a diagram of a neural network. At the bottom of the page are links for 'Sign in to a different account' and 'Create a new AWS account'. The footer contains the text '© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.' and a language selection dropdown set to 'English'.

**Test2.** I kept the ACCESS\_KEY, SECRET\_ACCESS\_KEY and BUCKET\_NAME secret, save it as an environment variable:

The screenshot shows the AWS Lambda console interface. On the left, a sidebar lists various configuration options: General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables (which is currently selected), Tags, VPC, Monitoring and operations tools, Concurrency, Asynchronous invocation, Code signing, and RDS databases. The main content area displays the 'FindNearestSchools' function. It includes a 'Function overview' section with tabs for 'Diagram' (selected) and 'Template'. Below this is a diagram showing the function's architecture. The 'Configuration' tab is active, showing the 'Environment variables' section. This section contains three entries:

Key	Value
ACCESS_KEY	AKIAKYKJT6L4CLFHAV4
BUCKET_NAME	iryna3bucket
SECRET_ACCESS_KEY	Gho8Ol6/3ZpHZ2T9QChj+5u3zwoQIUz5rV/5xfm3

All environment variables that are encrypted to prevent rogue access.

Unauthorized users can't use functions: GetLocationInfo(latitude, longitude, radius) and save\_a\_file(filedata, filename) without gain access to sensitive information stored in the solution, such as API keys and bucket name. In this case we got :

ValueError: Invalid API key provided.

```

< colab.research.google.com/drive/1fGx0jNBoj54wq87atbwG3QRdyFLjyEC#scrollTo=AwaSKQWnH9RX
Software SecurityProject.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[{"x": "def setAPI_key():
    key = input('Enter the API key: ')
    os.environ['API_KEY'] = key
    output.clear()

def GetlocationInfo(latitude, longitude, radius):
    gmaps = googlemaps.Client(os.environ.get('API_KEY'))
    string_shools = ''
    if not isinstance(latitude,(float)) or not isinstance(latitude,(int)):
        raise ValueError("Invalid latitude data type")
    if not isinstance(longitude,(float)) and not isinstance(longitude,(int)):
        raise TypeError("Invalid longitude data type")
    if not isinstance(radius,(float)) and not isinstance(radius,(int)):
        raise TypeError("Invalid radius data type")
    if latitude > 90.0 or latitude < -90.0:
        raise ValueError("Invalid latitude value")
    if longitude > 180.0 or longitude < -180.0:
        raise ValueError("Invalid longitude value")
    if radius <= 0:
        raise ValueError("Check the value of radius: is radius a non-negative?")
    try:
        places_result = gmaps.places_nearby(location=str(latitude) + ',' + str(longitude), radius = radius*1000, type = "school")
        for place in places_result['results']:
            string_shools = string_shools + str(latitude) + ',' + str(longitude) + ',' + str(radius) + ',' + str(place['name']) + '\n'
    except Exception as e:
        print("Error: " + str(e))
    return string_shools
setAPI_key()
print("test1")
result1 = GetlocationInfo(55.816555,-4.309890,4)
print(result1)
print("test2:")
result2 = GetlocationInfo(87.543255,0,10)
print(result2)

test1:
ValueError: Invalid latitude value
<ipython-input-1-28f45a759eac> in <cell line: 40>
      38 setAPI_key()
      39 print("test1")
--> 40 result1 = GetlocationInfo(55.816555,-4.309890,4)
      41 print(result1)
      42 print("test2:")

-> /usr/local/lib/python3.10/dist-packages/googlemaps/client.py in __init__(self, key, client_id, client_secret, timeout, connect_timeout, read_timeout, retry_timeout, requests_kwargs, queries_per_second, queries_per_minute, channel, retry_over_query_limit, experience_id, requests_session, base_url)
    142         if key and not key.startswith("AIza"):
    143             raise ValueError("Invalid API key provided.")
--> 144     self._client_id = client_id
    145     self._client_secret = client_secret
    146     if channel:
```

ValueError: Invalid API key provided.

```

< colab.research.google.com/drive/1fGx0jNBoj54wq87atbwG3QRdyFLjyEC#scrollTo=A3qjDG8vnaQ-
Software SecurityProject.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[{"x": "s3_client = get_S3_client()

def set_environment_variable_values():
    ACCESS_KEY = input("Please enter the AWS access key: ")
    SECRET_ACCESS_KEY = input("Please enter the AWS secret access key: ")
    BUCKET_NAME = input("Please enter the name of the bucket in S3: ")
    os.environ['ACCESS_KEY'] = ACCESS_KEY
    os.environ['SECRET_ACCESS_KEY'] = SECRET_ACCESS_KEY
    os.environ['BUCKET_NAME'] = BUCKET_NAME
    clear_output()
    return None

set_environment_variable_values()

import boto3

def get_S3_client():
    resource = boto3.client(
        's3',
        aws_access_key_id = os.environ.get('ACCESS_KEY'),
        aws_secret_access_key = os.environ.get('SECRET_ACCESS_KEY')
    )
    return resource

s3_client = get_S3_client()
print(s3_client)

def save_a_file(filedata, filename):
    # create a new file by uploading the data with the new filename
    response = s3_client.put_object(Bucket=os.environ.get('BUCKET_NAME'), Body=filedata, Key=filename)

data = GetlocationInfo(55.816555,-4.309890,4)
response = save_a_file(data, 'schools.csv')

data = GetlocationInfo(55.816555,-4.309890,4)
response = save_a_file(data, 'schools.csv')

-> /usr/local/lib/python3.10/dist-packages/googlemaps/client.py in __init__(self, key, client_id, client_secret, timeout, connect_timeout, read_timeout, retry_timeout, requests_kwargs, queries_per_second, queries_per_minute, channel, retry_over_query_limit, experience_id, requests_session, base_url)
    143         if key and not key.startswith("AIza"):
    144             raise ValueError("Invalid API key provided.")
--> 145     self._client_id = client_id
    146     self._client_secret = client_secret
    147     if channel:
```

ValueError: Invalid API key provided.

**Test3.** I try to access the endpoint in my browser by pasting the URL into the address bar

<https://5om9vjlyli.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchools>

I got the response {"message":"Forbidden"} - this is because the endpoint is protected by an API key.

```

← → ⌂ 5om9vjlyli.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchools
{"message": "Forbidden"}

```

## 2. Create and use a testing Plan:

- Make a plan for how you'll test the whole system thoroughly. This includes testing the local code (on your own computer) and the Lambda function (on AWS). Plan to use a 'test-driven' approach (create tests first and test your code repeatedly as you develop it).
- Create automated tests in your colab
- Create tests in Postman

## 3. Record tests in a table:

- Create a table listing all the tests you'll run. These tests should cover things like:
- Having no data in the request.
- Sending the wrong types of data.
- Missing data that's needed.
- Trying to run the code without proper authentication (like a password or key).
- And other potential issues.

Test no.	What is being tested?	Data (latitude, longitude, radius)	Expected outcome	Actual outcome	Notes
<b>COLAB TESTS</b>					
1.	Test <b>GetLocationInfo(latitude, longitude, radius)</b> With valid data	<b>GetLocationInfo(5.816555,-4.309890,4)</b>	55.816555, -4.30989, 4, Hutchesons' Grammar School 55.816555, -4.30989, 4, Eastwood High School 55.816555, -4.30989, 4, Glasgow Club Bellahouston 55.816555, -4.30989, 4, Bellahouston Academy 55.816555, -4.30989, 4, Pro-Soccer 55.816555, -4.30989, 4, St Vincent's Primary School 55.816555, -4.30989, 4, Cleeves Primary School 55.816555, -4.30989, 4, Thornliebank Primary School 55.816555, -4.30989, 4, Our Lady of the Mission Primary School 55.816555, -4.30989, 4, Hillpark Autism Unit 55.816555, -4.30989, 4, Woodland Outdoor Kindergartens 55.816555, -4.30989, 4, Kumon Maths & English 55.816555, -4.30989, 4, Carmichael Nursery School 55.816555, -4.30989, 4, Wendy King 55.816555, -4.30989, 4, Burnbrae Children's Centre 55.816555, -4.30989, 4, Zippytotz 55.816555, -4.30989, 4, Tinto Primary School 55.816555, -4.30989, 4, St Vincent's Autism Unit Co St Vincent's Primar	55.816555, -4.30989, 4, Hutchesons' Grammar School 55.816555, -4.30989, 4, Eastwood High School 55.816555, -4.30989, 4, Glasgow Club Bellahouston 55.816555, -4.30989, 4, Bellahouston Academy 55.816555, -4.30989, 4, Pro-Soccer 55.816555, -4.30989, 4, St Vincent's Primary School 55.816555, -4.30989, 4, Cleeves Primary School 55.816555, -4.30989, 4, Thornliebank Primary School 55.816555, -4.30989, 4, Our Lady of the Mission Primary School 55.816555, -4.30989, 4, Hillpark Autism Unit 55.816555, -4.30989, 4, Woodland Outdoor Kindergartens 55.816555, -4.30989, 4, Kumon Maths & English 55.816555, -4.30989, 4, Carmichael Nursery School 55.816555, -4.30989, 4, Wendy King 55.816555, -4.30989, 4, Burnbrae Children's Centre 55.816555, -4.30989, 4, Zippytotz 55.816555, -4.30989, 4, Tinto Primary School 55.816555, -4.30989, 4, St Vincent's Autism Unit Co St Vincent's Primar	This test works

2.	Test GetLocationInfo(latitude, longitude, radius) With valid data	<pre> data1 = GetLocationInfo(55.816555,-4.309890,4) response1 = save_a_file(data1,'schools.csv') data2 = GetLocationInfo(50.816555,-3.309890,7) response2 = save_a_file(data2,'schools.csv') data3 = GetLocationInfo(55.816555,-3.309890,9) response3 = save_a_file(data3,'schools.csv')  print(get_file(s3.client, 'schools.csv')) </pre>	55.816555, -4.30989, 4, Hutchiesons' Grammar School 55.816555, -4.30989, 4, Eastwood High School 55.816555, -4.30989, 4, Glasgow Club Bellahouston 55.816555, -4.30989, 4, Bellahouston Academy 55.816555, -4.30989, 4, Pro-Soccer 55.816555, -4.30989, 4, St Vincent's Primary School 55.816555, -4.30989, 4, Queen Elizabeth Primary School 55.816555, -4.30989, 4, Our Lady of the Mission Primary School 55.816555, -4.30989, 4, Hillpark Autism Unit 55.816555, -4.30989, 4, Fotheringay Centre 55.816555, -4.30989, 4, Animal Man Parties 55.816555, -4.30989, 4, The Little Acorn Daycare & Kindergartens 55.816555, -4.30989, 4, Kumon Maths & English 55.816555, -4.30989, 4, Carmichael Nursery School 55.816555, -4.30989, 4, Wendy King 55.816555, -4.30989, 4, Burnbrae Children's Centre 55.816555, -4.30989, 4, Zippytots 55.816555, -4.30989, 4, St Vincent's Autism Unit Co St Vincent's Primary 50.816555, -3.30989, 7, Payhembury C Of E Primary School 50.816555, -3.30989, 7, Feniton Primary School 50.816555, -3.30989, 7, Powerboat & RIB 50.816555, -3.30989, 7, Clyst Hydon Primary School 50.816555, -3.30989, 7, The Forest School at Escot 50.816555, -3.30989, 7, Broadheury Church of England Primary School 50.816555, -3.30989, 7, Awliscombe Primary School 50.816555, -3.30989, 7, Acorns Stars Preschool 50.816555, -3.30989, 7, Kentisbeare C Of E Primary School 50.816555, -3.30989, 7, Nanny Bears Nursery and Pre-school 50.816555, -3.30989, 7, Whimble Preschool Playgroup 50.816555, -3.30989, 7, Whimble Primary School 50.816555, -3.30989, 7, A B C Day Nursery 50.816555, -3.30989, 9, North Walls Primary School 50.816555, -3.30989, 9, North Walls Junior High School 50.816555, -3.30989, 9, Little wards	55.816555, -4.30989, 4, Hutchiesons' Grammar School 55.816555, -4.30989, 4, Eastwood High School 55.816555, -4.30989, 4, Glasgow Club Bellahouston 55.816555, -4.30989, 4, Bellahouston Academy 55.816555, -4.30989, 4, Pro-Soccer 55.816555, -4.30989, 4, St Vincent's Primary School 55.816555, -4.30989, 4, Queen Elizabeth Primary School 55.816555, -4.30989, 4, Our Lady of the Mission Primary School 55.816555, -4.30989, 4, Hillpark Autism Unit 55.816555, -4.30989, 4, Fotheringay Centre 55.816555, -4.30989, 4, Animal Man Parties 55.816555, -4.30989, 4, The Little Acorn Daycare & Kindergartens 55.816555, -4.30989, 4, Kumon Maths & English 55.816555, -4.30989, 4, Carmichael Nursery School 55.816555, -4.30989, 4, Wendy King 55.816555, -4.30989, 4, Burnbrae Children's Centre 55.816555, -4.30989, 4, Zippytots 55.816555, -4.30989, 4, St Vincent's Autism Unit Co St Vincent's Primary 50.816555, -3.30989, 7, Payhembury C Of E Primary School 50.816555, -3.30989, 7, Feniton Primary School 50.816555, -3.30989, 7, Powerboat & RIB 50.816555, -3.30989, 7, Clyst Hydon Primary School 50.816555, -3.30989, 7, The Forest School at Escot 50.816555, -3.30989, 7, Broadheury Church of England Primary School 50.816555, -3.30989, 7, The Forest School at Escot 50.816555, -3.30989, 7, Awliscombe Primary School 50.816555, -3.30989, 7, Acorns Stars Preschool 50.816555, -3.30989, 7, Kentisbeare C Of E Primary School 50.816555, -3.30989, 7, Nanny Bears Nursery and Pre-school 50.816555, -3.30989, 7, Whimble Preschool Playgroup 50.816555, -3.30989, 7, Whimble Primary School 50.816555, -3.30989, 7, A B C Day Nursery 50.816555, -3.30989, 9, North Walls Primary School 50.816555, -3.30989, 9, North Walls Junior High School 50.816555, -3.30989, 9, Little wards	This test works
			COLAB UNITTESTS		
3.	Test GetLocationInfo (latitude, longitude, radius) With valid data	(55.816555,- 4.309890,4) GetLocationInfo(5 5.816555,- 4.309890,4)[51:95]	'55.816555, -4.30989, 4, Eastwood High School'	'55.816555, -4.30989, 4, Eastwood High School'	This test works
4.	Test GetLocationInfo (latitude, longitude, radius) With valid data	'55.816555, - 4.30989, 4, Tinto Primary School',GetLocatio nInfo(55.816555,- 4.309890,4)[-118:- 74]	'55.816555, -4.30989, 4, Tinto Primary School'	'55.816555, -4.30989, 4, Tinto Primary School'	This test works
5.	Test GetLocationInfo (latitude, longitude, radius) with an incorrect latitude data type	'abc',-4.309890,4	TypeError	TypeError	This test works
6.	Test GetLocationInfo (latitude, longitude, radius) with an incorrect latitude data	'True',-4.309890,4	TypeError	TypeError	This test works

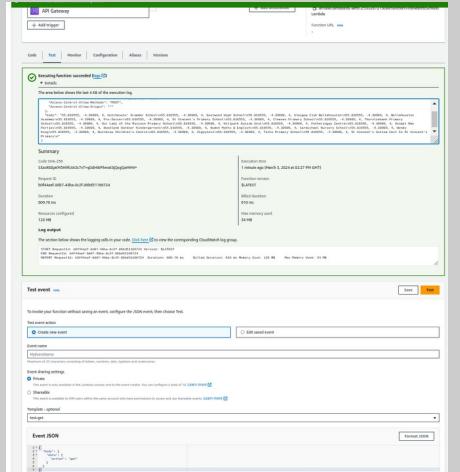
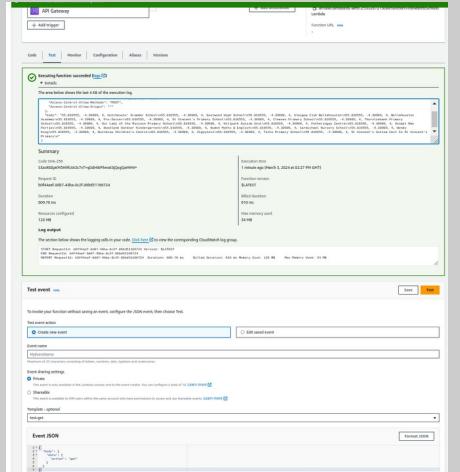
	type				
7.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect longitude data type	55.816555,'abc',4	TypeError	TypeError	This test works
8.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect longitude data type	55.816555,'False', 4	TypeError	TypeError	This test works
9.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect radius data type	55.816555,- 4.309890,'abc'	TypeError	TypeError	This test works
10.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect radius data type	55.67,- 4.309890,'True'	TypeError	TypeError	This test works
11.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect longitude data	69.999995,187.40, 5	ValueError	ValueError	This test works

	value				
12.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect longitude data value	-79.999995,- 189.230,5	ValueError	ValueError	This test works
13.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect latitude data value	99.999995,0,5	ValueError	ValueError	This test works
14.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect latitude data value	-99.999995,0,5	ValueError	ValueError	This test works
15.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect radius value	-69.999995,0,-5	ValueError	ValueError	This test works
16.	Test GetLocationInfo(latitude, longitude, radius)	87.543255,0,10	"	"	This test works
17.	Test delete_a_scho	'Glasgow Club	'55.816555, -4.30989, 4,	'55.816555, -4.30989, 4,	This test

	ol(filedata, name) with an correct data	Bellahouston' delete_a_school(GetLocationInfo(55.816555,-4.309890,4),'Glasgow Club Bellahouston')[96:140]	Bellahouston Academy'	Bellahouston Academy'	works
18.	Test delete_a_school(filedata,name) with an incorrect filedata data type	12345,'Zippytotz'	TypeError	TypeError	This test works
19.	Test delete_a_school(filedata,name) with an incorrect name data type	GetLocationInfo(55.816555,-4.309890,4),99999	TypeError	TypeError	This test works
20.	Test delete_a_school(filedata,name) with an incorrect filedata value	','Zippytotz'	ValueError	ValueError	This test works
21.	Test save_a_file(file data) with an incorrect filename type	save_a_file(123)	TypeError	TypeError	This test works
21 a.	Test save_a_file(file data) with an correct data	save_a_file(result1).statusCode	200	200	This test works
21 b.	Test save_a_file(file data) with an correct data	save_a_file(result1).text	"Schools saved"	"Schools saved"	This test works
22.	Test get_file(s3_clie	s3_client,'abc'	ValueError	ValueError	This test

	nt, filename) with an incorrect filename value				work s
23.	Test get_file() with an correct data in Colab	get_a_file()	<Response [200]>		This test work s
23 a	Test get_file() with an correct data in Colab 2 sets	get_a_file()	<Response [200]>		This test work s
23 b	Test get_file() with an correct data in Colab 3 sets	get_a_file()	<Response [200]>		This test work s
23c	Test get_file() with an correct data in Colab all data	get_a_file()	<Response [200]>		
24.	Test get_file() with an correct data in Colab unitests	get_file(s3_client, 'schools.csv') [51:95]	'55.816555, -4.30989, 4, Eastwood High School'	'55.816555, -4.30989, 4, Eastwood High School'	This test work s
25.	Test get_file() with an correct data in Colab unitests	get_file(s3_client, 'schools.csv')[ -37:- 1]	'58.816555, -3.30989, 9, Little wards'	'58.816555, -3.30989, 9, Little wards'	This test work s

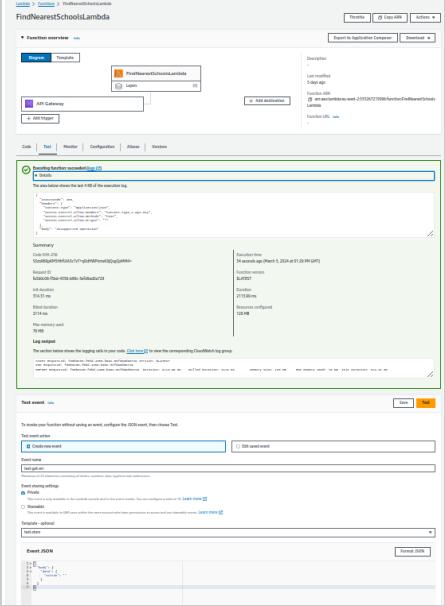
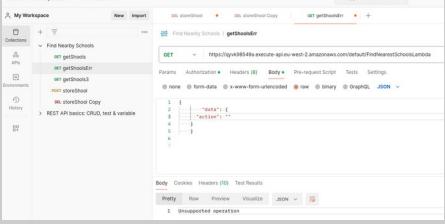
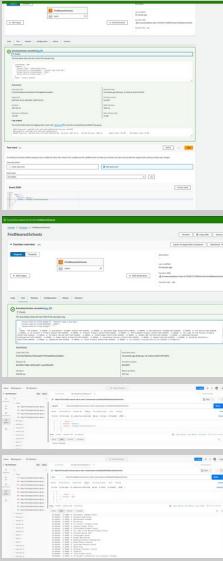
## AWS AND POSTMAN TESTS

26. Test AWS Lambda return a list of all schools in the database or file on request with valid key	<pre>{   "body": {     "data": {       "action": "get"     }   } }</pre> <p>"55.816555, -4.30989, 4,  Hutchesons' Grammar School\n55.816555, -4.30989, 4,  Eastwood High School\n55.816555, -4.30989, 4,  Glasgow Club Bellahouston\n55.816555, -4.30989, 4,  Bellahouston Academy\n55.816555, -4.30989, 4, Pro-Soccer\n55.816555, -4.30989, 4,  St Vincent's Primary School\n55.816555, -4.30989, 4, Cleeves Primary School\n55.816555, -4.30989, 4, Thornliebank Primary School\n55.816555, -4.30989, 4,  Our Lady of the Mission Primary School\n55.816555, -4.30989, 4,  Hillpark Autism Unit\n55.816555, -4.30989, 4, Fotheringay Centre\n55.816555, -4.30989, 4, Animal Man Parties\n55.816555, -4.30989, 4, Woodland Outdoor Kindergartens\n55.816555, -4.30989, 4, Kumon Maths &amp; English\n55.816555, -4.30989, 4, Carmichael Nursery School\n55.816555, -4.30989, 4, Wendy King\n55.816555, -4.30989, 4, Burnbrae Children's Centre\n55.816555, -4.30989, 4, Zippytotz\n55.816555, -4.30989, 4, Tinto Primary School\n55.816555, -4.30989, 4, St Vincent's Autism Unit Co St Vincent's Primary\n"</p> <td data-bbox="1024 215 1583 1796">  </td> <td data-bbox="1500 932 1583 1796">This test works</td>		This test works
POSTMAN TESTS	<pre>{   "body": {     "data": {       "action": "get"     }   } }</pre> <p>55.816555, -4.30989, 4,  Hutchesons' Grammar School 55.816555, -4.30989, 4, Eastwood High School  55.816555, -4.30989, 4,</p> <td data-bbox="1024 1796 1583 2077">  </td> <td data-bbox="1500 1796 1583 2077"></td>		

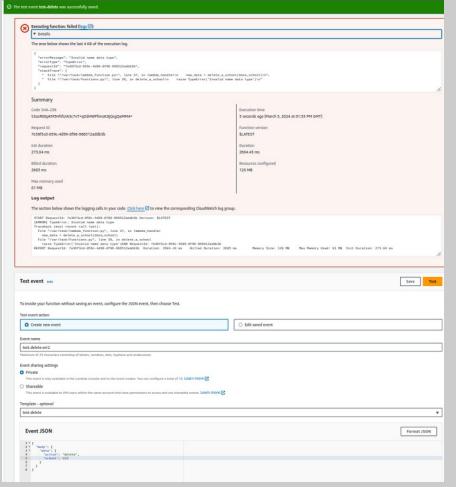
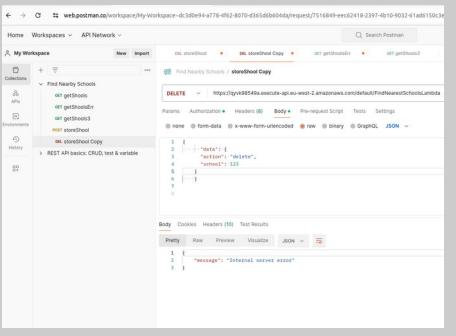
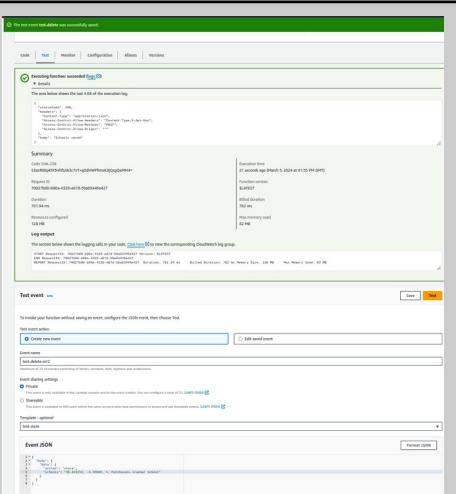
Glasgow Club Bellahouston  
55.816555, -4.30989, 4,  
Bellahouston Academy  
55.816555, -4.30989, 4, Pro-  
Soccer 55.816555, -4.30989,  
4, St Vincent's Primary  
School 55.816555, -4.30989,  
4, Cleeves Primary School  
55.816555, -4.30989, 4,  
Thornliebank Primary  
School 55.816555, -4.30989,  
4, Our Lady of the Mission  
Primary School 55.816555, -  
4.30989, 4, Hillpark Autism  
Unit 55.816555, -4.30989, 4,  
Fotheringay Centre  
55.816555, -4.30989, 4,  
Animal Man Parties  
55.816555, -4.30989, 4,  
Woodland Outdoor  
Kindergartens 55.816555, -  
4.30989, 4, Kumon Maths &  
English 55.816555, -  
4.30989, 4, Carmichael  
Nursery School 55.816555, -  
4.30989, 4, Wendy King  
55.816555, -4.30989, 4,  
Burnbrae Children's Centre  
55.816555, -4.30989, 4,  
Zippytotz 55.816555, -  
4.30989, 4, Tinto Primary  
School 55.816555, -4.30989,  
4, St Vincent's Autism Unit  
Co St Vincent's Primary

27  
a.

<p>Test AWS Lambda return a list of all schools (three list of data) in the database or file on request with valid key</p>	<pre>{   "body": {     "data": {       "action": "get"     }   } }</pre> <p>"58.816555, -3.30989, 9, North Walls Primary School\\n58.816555, -3.30989, 9, North Walls Junior High School\\n58.816555, -3.30989, 9, Little wards\\n50.816555, -3.30989, 7, Payhembury C Of E Primary School\\n50.816555, -3.30989, 7, Feniton Primary School\\n50.816555, -3.30989, 7, Powerboat &amp; RIB\\n50.816555, -3.30989, 7, Clyst Hydon Primary School\\n50.816555, -3.30989, 7, Plymtree Pre-school\\n50.816555, -3.30989, 7, Plymtree C Of E Primary School\\n50.816555, -3.30989, 7, Broadhembury Church of England Primary School\\n50.816555, -3.30989, 7, The Forest School at Escot\\n50.816555, -3.30989, 7, Awliscombe Primary School\\n50.816555, -3.30989, 7, Accent Studios\\n50.816555, -3.30989, 7, Kentisbeare C Of E Primary School\\n50.816555, -3.30989, 7, Nanny Bears Nursery and Pre-school\\n50.816555, -3.30989, 7, Whimple Preschool Playgroup\\n50.816555, -3.30989, 7, Whimple Primary School\\n50.816555, -3.30989, 7, A B C Day Nursery\\n58.816555, -3.30989, 9, North Walls Primary School\\n58.816555, -3.30989, 9, North Walls Junior High School\\n58.816555, -3.30989, 9, Little wards\\n"</p>		<p>This test works</p>
<p>POSTMAN TESTS</p>	<pre>{   "body": {     "data": {       "action": "get"     }   } }</pre>		<p>This test works</p>

			This test works
28.	<p>Test AWS Lambda return a list of all schools in the database or file on request with invalid key</p> <pre>{   "body": {     "data": {       "action": ""     }   } }</pre>	"Unsupported operation"	
	<p>POSTMAN TESTS</p> <pre>{   "body": {     "data": {       "action": ""     }   } }</pre>	"Unsupported operation"	
29.	<p>Test AWS Lambda will accept a name of a school to delete from the file on S3 with invalid key</p> <pre>{   "body": {     "data": {       "action": "delete",       "school": "Glasgow Club Bellahouston"     }   } }</pre>	"School deleted"	

	POSTMAN TESTS	<pre>{   "body": {     "data": {       "action": "delete",       "school": "Glasgow Club Bellahouston"     }   } }</pre>	"School deleted"		This test works
30.	Test AWS Lambda will accept a name of a school to delete from the file on S3 with invalid key	<pre>{   "body": {     "data": {       "action": "",       "school": "Glasgow Club Bellahouston"     }   } }</pre>	"Unsupported operation"		This test works
	POSTMAN TESTS	<pre>{   "body": {     "data": {       "action": "",       "school": "Glasgow Club Bellahouston"     }   } }</pre>	"Unsupported operation"		
31.	Test AWS Lambda will accept a	<pre>{   "body": {     "data": {       "action": ""     }   } }</pre>	"errorMessage": "Invalid name data type",	<pre>"errorMessage": "Invalid name data type",</pre>	This test shows

	<pre> name of a school to delete from the file on S3 with invalid type key </pre>	<pre> "data": {   "action": "delete",   "school": 123 } } } </pre>	<pre> "errorType": "TypeError", </pre> 
	<pre> POSTMAN TESTS </pre>	<pre> {   "body": {     "data": {       "action": "delete",       "school": 123     }   } } </pre>	<pre> "errorMessage": "Invalid name data type", "errorType": "TypeError", </pre>  <p>"Internal server error"</p>
32.	<pre> Test AWS Lambda store data in s3 </pre>	<pre> {   "body": {     "data": {       "action": "store",       "schools": "55.816555, -4.30989, 4, Hutchesons Grammar School"     }   } } </pre>	<pre> "Schools saved" </pre>  <p>"Schools saved"</p>

POSTMAN TESTS	<pre>{   "body": {     "data": {       "action": "store",       "schools": "55.816555, -4.30989, 4, Hutchesons Grammar School"     }   } }</pre>	"Schools saved"		This test works
POSTMAN TESTS	<pre>{   "body": {     "data": {       "action": "store",       "schools": "55.816555, -4.30989, 4, ABCD School"     }   } }</pre>	"Schools saved"		This test works

#### 4. Test in Steps:

- Start by testing the code on your own computer without sending data to the API. Then, test it again but this time, actually sending data.

Unitests def GetLocationInfo(latitude, longitude, radius):

```
import unittest
class TestsGetLocationInfo(unittest.TestCase):
    # test with an correct data
    def test_valid_data1(self):
        self.assertEqual('55.816555, -4.30989, 4, Eastwood High
School',GetLocationInfo(55.816555,-4.309890,4)[51:95])
    # test with an correct data
    def test_valid_data2(self):
```

```
    self.assertEqual('55.816555, -4.30989, 4, Tinto Primary
School',GetLocationInfo(55.816555,-4.309890,4)[-118:-74])
    # test with an correct data
def test_valid_data3(self):
    self.assertEqual("",GetLocationInfo(87.543255,0,10))
# test with an incorrect latitude data type
def test_GetLocationInfo_data_typeerror_1(self):
    with self.assertRaises(TypeError):
        GetLocationInfo('abc',-4.309890,4)
# test with an incorrect longitude data type
def test_GetLocationInfo_data_typeerror_2(self):
    with self.assertRaises(TypeError):
        GetLocationInfo('True',-4.309890,4)
# test with an incorrect radius data type
def test_GetLocationInfo_data_typeerror_3(self):
    with self.assertRaises(TypeError):
        GetLocationInfo(55.816555,'abc',4)
# test with an incorrect longitude data type
def test_GetLocationInfo_data_typeerror_4(self):
    with self.assertRaises(TypeError):
        GetLocationInfo(55.816555,'False',4)
# test with an incorrect radius data type
def test_GetLocationInfo_data_typeerror_5(self):
    with self.assertRaises(TypeError):
        GetLocationInfo(55.816555,-4.309890,'abc')
# test with an incorrect radius data type
def test_GetLocationInfo_data_typeerror_6(self):
    with self.assertRaises(TypeError):
        GetLocationInfo(55.67,-4.309890,'True')
# test with an incorrect longitude data value
def test_valueerror1(self):
    with self.assertRaises(ValueError):
        GetLocationInfo(69.999995,187.40,5)
# test with an incorrect longitude data value
def test_valueerror2(self):
    with self.assertRaises(ValueError):
        GetLocationInfo(-79.999995,-189.230,5)
# test with an incorrect latitude data value
```

```
def test_valueerror3(self):
    with self.assertRaises(ValueError):
        GetLocationInfo(99.999995,0,5)
# test with an incorrect latitude data value
def test_valueerror4(self):
    with self.assertRaises(ValueError):
        GetLocationInfo(-99.999995,0,5)

# test with an incorrect radius value
def test_valueerror5(self):
    with self.assertRaises(ValueError):
        GetLocationInfo(-69.999995,0,-5)

if __name__ == '__main__':
    s = unittest.TestLoader().loadTestsFromTestCase(TestsGetLocationInfo)
    unittest.TextTestRunner().run(s)
```

.....  
-----  
Ran 14 tests in 0.393s

OK

Software SecurityProject.ipynb

```
+ Code + Text
import unittest
class TestsGetLocationInfo(unittest.TestCase):
    # test with an correct data
    def test_valid_data(self):
        self.assertEqual('55.816555, -4.309890, 4, Eastwood High School', GetLocationInfo(55.816555, -4.309890, 4)[51:95])
    # test with an correct data
    def test_valid_data2(self):
        self.assertEqual('55.816555, -4.309890, 4, Tinto Primary School', GetLocationInfo(55.816555, -4.309890, 4)[-118:-74])
    # test with an correct data
    def test_valid_data3(self):
        self.assertEqual('55.816555, -4.309890, 4, abc', GetLocationInfo(55.816555, -4.309890, 4)[-118:-74])
    # test with an incorrect latitude data type
    def test_GetLocationInfo_data_typeerror_1(self):
        with self.assertRaises(TypeError):
            GetLocationInfo('abc', -4.309890, 4)
    # test with an incorrect longitude data type
    def test_GetLocationInfo_data_typeerror_2(self):
        with self.assertRaises(TypeError):
            GetLocationInfo('True', -4.309890, 4)
    # test with an incorrect radius data type
    def test_GetLocationInfo_data_typeerror_3(self):
        with self.assertRaises(TypeError):
            GetLocationInfo(55.816555, 'abc', 4)
    # test with an incorrect radius data type
    def test_GetLocationInfo_data_typeerror_4(self):
        with self.assertRaises(TypeError):
            GetLocationInfo(55.816555, 'False', 4)
    # test with an incorrect radius data type
    def test_GetLocationInfo_data_typeerror_5(self):
        with self.assertRaises(TypeError):
            GetLocationInfo(55.816555, -4.309890, 'abc')
    # test with an incorrect radius data type
    def test_GetLocationInfo_data_typeerror_6(self):
        with self.assertRaises(TypeError):
            GetLocationInfo(55.67, -4.309890, 'True')
    # test with an incorrect longitude data value
    def test_valueerror1(self):
        with self.assertRaises(ValueError):
            GetLocationInfo(55.999995, 187.40, 5)
    # test with an incorrect longitude data value
    def test_valueerror2(self):
        with self.assertRaises(ValueError):
            GetLocationInfo(-79.999995, -189.230, 5)
    # test with an incorrect longitude data value
    def test_valueerror3(self):
        with self.assertRaises(ValueError):
            GetLocationInfo(55.999995, 0, 5)
    # test with an incorrect longitude data value
    def test_valueerror4(self):
        with self.assertRaises(ValueError):
            GetLocationInfo(-99.999995, 0, 5)
    # test with an incorrect radius value
    def test_valueerror5(self):
        with self.assertRaises(ValueError):
            GetLocationInfo(-69.999995, 0, -5)

if __name__ == '__main__':
    s = unittest.TestLoader().loadTestsFromTestCase(TestsGetLocationInfo)
    unittest.TextTestRunner().run(s)
```

Software SecurityProject.ipynb

```
+ Code + Text
def test_GetLocationInfo_data_typeerror_4(self):
    with self.assertRaises(TypeError):
        GetLocationInfo(55.816555, 'False', 4)
# test with an incorrect radius data type
def test_GetLocationInfo_data_typeerror_5(self):
    with self.assertRaises(TypeError):
        GetLocationInfo(55.816555, -4.309890, 'abc')
# test with an incorrect radius data type
def test_GetLocationInfo_data_typeerror_6(self):
    with self.assertRaises(TypeError):
        GetLocationInfo(55.67, -4.309890, 'True')
# test with an incorrect longitude data value
def test_valueerror1(self):
    with self.assertRaises(ValueError):
        GetLocationInfo(59.999995, 187.40, 5)
# test with an incorrect longitude data value
def test_valueerror2(self):
    with self.assertRaises(ValueError):
        GetLocationInfo(-79.999995, -189.230, 5)
# test with an incorrect longitude data value
def test_valueerror3(self):
    with self.assertRaises(ValueError):
        GetLocationInfo(99.999995, 0, 5)
# test with an incorrect longitude data value
def test_valueerror4(self):
    with self.assertRaises(ValueError):
        GetLocationInfo(-99.999995, 0, 5)

# test with an incorrect radius value
def test_valueerror5(self):
    with self.assertRaises(ValueError):
        GetLocationInfo(-69.999995, 0, -5)

if __name__ == '__main__':
    s = unittest.TestLoader().loadTestsFromTestCase(TestsGetLocationInfo)
    unittest.TextTestRunner().run(s)
```

Ran 14 tests in 0.393s

OK

✓ 0s completed at 23:47

```
Unitests def delete_a_school(filedata,name):
```

```
import unittest
```

```
class TestsDeleteSchool(unittest.TestCase):
```

```
# test with an correct data
```

```
def test_valid_data1(self):
```

```
    self.assertEqual('55.816555, -4.30989, 4, Bellahouston
```

```
Academy',delete_a_school(GetLocationInfo(55.816555,-4.309890,4),'Glasgow Club
```

```
Bellahouston')[96:140])
```

```
# test with an incorrect filedatal data type
```

```
def test_DeleteSchool_data_typeerror_1(self):
```

```
    with self.assertRaises(TypeError):
```

```
        delete_a_school(12345,'Zippytotz')
```

```
# test with an incorrect name data type
```

```
def test_DeleteSchool_data_typeerror_2(self):
```

```
    with self.assertRaises(TypeError):
```

```
        delete_a_school(GetLocationInfo(55.816555,-4.309890,4),99999)
```

```
# test with an incorrect filedatal value
```

```
def test_valueerror5(self):
```

```
    with self.assertRaises(ValueError):
```

```
        delete_a_school("','Zippytotz")
```

```
if __name__ == '__main__':
```

```
s = unittest.TestLoader().loadTestsFromTestCase(TestsDeleteSchool)
```

```
unittest.TextTestRunner().run(s)
```

```
Ran 4 tests in 0.581s
```

```
OK
```

```

06 import unittest
class TestsDeleteSchool(unittest.TestCase):

    # test with an correct data
    def test_valid_data(self):
        self.assertEqual('55.816555, -4.30989, 4, Bellahouston Academy', delete_a_school(GetLocationInfo(55.816555,-4.309890,4), 'Glasgow Club Bellahouston'))

    # test with an incorrect filedatal data type
    def test_DeleteSchool_data_typeerror_1(self):
        with self.assertRaises(TypeError):
            delete_a_school(12345,'Zippytotz')
    # test with an incorrect name data type
    def test_DeleteSchool_data_typeerror_2(self):
        with self.assertRaises(TypeError):
            delete_a_school(GetLocationInfo(55.816555,-4.309890,4),99999)

    # test with an incorrect filedatal value
    def test_valueerror5(self):
        with self.assertRaises(ValueError):
            delete_a_school('', 'Zippytotz')

if __name__ == '__main__':
    s = unittest.TestLoader().loadTestsFromTestCase(TestsDeleteSchool)
    unittest.TextTestRunner().run(s)
    ...
Ran 4 tests in 0.581s
OK
]

```

### Unitests def save\_a\_file(filedata, filename):

```

import unittest
class TestsSaveFile(unittest.TestCase):

    # test with an correct data
    def test_valid_data1(self):
        self.assertEqual(200,save_a_file(result1).statusCode)

    # test with an correct data
    def test_valid_data1(self):
        self.assertEqual("Schools saved",save_a_file(result1).text)

    # test with an incorrect filename value
    def test_valueerror(self):
        with self.assertRaises(TypeError):
            save_a_file(123)

if __name__ == '__main__':

    s = unittest.TestLoader().loadTestsFromTestCase(TestsSaveFile)
    unittest.TextTestRunner().run(s)

```

CO Copy of Software SecurityProject.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

10s

{x}

import unittest

class TestsSaveFile(unittest.TestCase):

# test with an correct data

def test\_valid\_data(self):

self.assertEqual(200, save\_a\_file(result1).statusCode)

# test with an correct data

def test\_valid\_data1(self):

self.assertEqual("Schools saved", save\_a\_file(result1).text)

# test with an incorrect filename value

def test\_valueerror(self):

with self.assertRaises(TypeError):

save\_a\_file(123)

if \_\_name\_\_ == '\_\_main\_\_':

s = unittest.TestLoader().loadTestsFromTestCase(TestsSaveFile)

unittest.TextTestRunner().run(s)

Enter the API key so that it isn't stored in the notebook: lVjE3I0uUdaU72VgUHws45je0sEeEUurhpWKTSi9

Ran 2 tests in 9.801s

OK

Unitests def get\_file(s3\_client, filename):

```
import unittest
class TestsGetFile(unittest.TestCase):

# test with an correct data
def test_valid_data1(self):
    self.assertEqual('55.816555, -4.30989, 4, Eastwood High School',get_a_file()[51:95])
if __name__ == '__main__':

s = unittest.TestLoader().loadTestsFromTestCase(TestsGetFile)
unittest.TextTestRunner().run(s)
```

```

import unittest
class TestsGetFile(unittest.TestCase):

    # test with an correct data
    def test_valid_data(self):
        self.assertEqual('55.816555, -4.30989, 4, Eastwood High School', get_a_file()[51:95])

if __name__ == '__main__':
    s = unittest.TestLoader().loadTestsFromTestCase(TestsGetFile)
    unittest.TextTestRunner().run(s)

Enter the API key so that it isn't stored in the notebook: lVjE3I0uUdaU72VgUhws45jeOsEeUurhpWKTSi9
.
.
.
Ran 1 test in 47.592s
OK

```

## 5. Test the Lambda Function:

- Test the Lambda function at every stage. This means checking if it works when it's triggered by the API Gateway and if it can read and write files in S3.

### POSTMAN GET

<https://qvvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda>

The screenshot shows the Postman interface with a collection named "Find Nearby Schools". A GET request is selected with the URL: <https://qvvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda>. The request body is set to raw JSON:

```

1 {
2   .... "data": [
3     .... "action": "get"
4     .... ]
5   ....
6 }
7

```

The response status is 200 OK, and the response body is displayed as:

```

[{"lat": 55.816555, "lon": -4.30989, "name": "Zippytoz", "address": "Tinto Primary School"}, {"lat": 55.816555, "lon": -4.30989, "name": "St Vincent's Autism Unit Co St Vincent's Primary", "address": "Payhembury C Of E Primary School"}, {"lat": 55.816555, "lon": -3.30989, "name": "Feniton Primary School", "address": "Powerboat & RIB"}, {"lat": 55.816555, "lon": -3.30989, "name": "Clyst Hydon Primary School", "address": "Plymtree Pre-school"}, {"lat": 55.816555, "lon": -3.30989, "name": "Plymtree C Of E Primary School", "address": "The Forest School at EBBOT"}, {"lat": 55.816555, "lon": -3.30989, "name": "Awliscombe Primary School", "address": "Accent Studios"}, {"lat": 55.816555, "lon": -3.30989, "name": "Kentsisbeare C Of E Primary School", "address": "Nanny Bears Nursery and Pre-school"}, {"lat": 55.816555, "lon": -3.30989, "name": "Whimble Preschool Playgroup", "address": ""}]

```

Home Workspaces API Network

My Workspace

Collections APIs Environments History REST API basics: CRUD, test & variable

Find Nearby Schools / getSchools

GET https://qyv98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda

Params Authorization Headers (8) Body Pre-request Script Tests Settings

```
1 pm.test("response must be valid and have a body", function () {  
2   pm.response.to.be.ok;  
3   pm.response.to.be.withBody;  
4 });  
  
5  
6 pm.test("response should be okay to process", function () {  
7   pm.response.to.not.be.error;  
8   pm.expect(pm.response.text()).to.include("55.816555, -4.30989, 4, Eastwood High School");  
9   pm.expect(pm.response.text()).to.include("50.816555, -3.30989, 7, Paynhambury C Of E Primary School");  
10  pm.expect(pm.response.text()).to.include("58.816555, -3.30989, 9, Little wards");  
11  pm.response.to.not.have.jsonBody("error");  
12 });  
13  
14 pm.test("Status test", function () {  
15   pm.response.to.have.status(200);  
16 });  
17
```

Body Cookies Headers (10) Test Results (3/3)

Status: 200 OK Time: 4.89s Size: 2.27 KB Save as example

Postbot Runner Auto-select agent Cookies Trash

Home Workspaces API Network

My Workspace

Collections APIs Environments History REST API basics: CRUD, test & variable

Find Nearby Schools / getSchools

GET https://qyv98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda

Params Authorization Headers (8) Body Pre-request Script Tests Settings

```
1 pm.test("response must be valid and have a body", function () {  
2   pm.response.to.be.ok;  
3   pm.response.to.be.withBody;  
4 });  
  
5  
6 pm.test("response should be okay to process", function () {  
7   pm.response.to.not.be.error;  
8   pm.expect(pm.response.text()).to.include("55.816555, -4.30989, 4, Eastwood High School");  
9   pm.expect(pm.response.text()).to.include("50.816555, -3.30989, 7, Paynhambury C Of E Primary School");  
10  pm.expect(pm.response.text()).to.include("58.816555, -3.30989, 9, Little wards");  
11  pm.response.to.not.have.jsonBody("error");  
12 });  
13  
14 pm.test("Status test", function () {  
15   pm.response.to.have.status(200);  
16 });  
17
```

Body Cookies Headers (10) Test Results (3/3)

All Passed Skipped Failed C

PASS response must be valid and have a body

PASS response should be okay to process

PASS Status test

Status: 200 OK Time: 4.89s Size: 2.27 KB Save as example

Postbot Runner Auto-select agent Cookies Trash

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Collections, APIs, Environments, History, and a plus sign for creating new items. The main area displays a collection named "My Workspace". Inside this collection, there's a folder named "Find Nearby Schools" containing several API endpoints:

- GET getShools**
- GET getShoolsErr**
- POST storeShool**
- DEL deleteShoolErr2**
- DEL deleteShoolErr1**
- DEL deleteShool**

Below the collection list, there's a link to "REST API basics: CRUD, test & variable".

The main workspace shows a specific API endpoint: **Find Nearby Schools / getShoolsErr**. The method is set to **GET**, and the URL is <https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchools>. The "Body" tab is selected, showing a JSON payload:

```
1 {  
2   "data": [{}],  
3   "action": "getShoolsErr"  
4 }  
5  
6  
7
```

Below the body, the "Test Results" tab is active, showing 1/1 test results. The status is **PASS** for the test "Status test".

The screenshot shows the Postman application interface. On the left, there's a sidebar with icons for Collections, APIs, Environments, History, and a plus sign for new items. The main area displays a collection named "My Workspace". A specific API endpoint, "Find Nearby Schools / getSchoolsErr", is selected. The request method is "GET", and the URL is "https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchools". The "Tests" tab is active, containing a script:

```
1 pm.test("Status test", function () {  
2     pm.response.to.have.status(400);  
3 });  
4 
```

Below the tests, the "Test Results" section shows 1/1 result, all of which are "Passed".

## POSTMAN POST

<https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda>

The screenshot shows the Postman interface with a collection named "Find Nearby Schools". A POST request is selected for the "storeSchool" endpoint. The "Tests" tab contains a JavaScript test script:

```
1 pm.test("Status test", function () {
2     pm.response.to.have.status(200);
3 });
4
5 pm.test("response should be okay to process", function () {
6     pm.response.to.not.be.error();
7     pm.response.to.not.have.jsonBody("error");
8     pm.expect(pm.response.text()).to.include("Schools saved");
9 });
10
11 pm.test("response must be valid and have a body", function () {
12     pm.response.to.be.ok;
13     pm.response.to.be.withBody;
14 });
```

The response status is 200 OK, time is 1343 ms, and size is 469 B.

The screenshot shows the Postman interface with a collection named "Find Nearby Schools". A POST request is selected for the "storeSchool" endpoint. The "Body" tab shows a JSON payload:

```
1 {
2     "data": {
3         "action": "store",
4         "schools": "55.816555, -4.39989, 4, ABCD School"
5     }
6 }
```

The response status is 200 OK, time is 1343 ms, and size is 469 B.

The screenshot shows the Postman interface with a collection named "Find Nearby Schools". A GET request is selected for the "getSchools" endpoint. The "Body" tab shows the response data, which is a list of school locations:

```
55.816555, -4.39989, 4, Pro-Soccer
55.816555, -4.39989, 4, St Vincent's Primary School
55.816555, -4.39989, 4, Clever Primary School
55.816555, -4.39989, 4, Thornliebank Primary School
55.816555, -4.39989, 4, Our Lady of the Mission Primary School
55.816555, -4.39989, 4, Hillpark Autism Unit
55.816555, -4.39989, 4, Fotheringay Centre
55.816555, -4.39989, 4, Woodland Outreach Kindergartens
55.816555, -4.39989, 4, Kumon Maths & English
55.816555, -4.39989, 4, Camichael Nursey School
55.816555, -4.39989, 4, Wendy King
55.816555, -4.39989, 4, Burnbrae Children's Centre
55.816555, -4.39989, 4, Zippytots
55.816555, -4.39989, 4, Tinto Primary School
55.816555, -4.39989, 4, St Vincent's Autism Unit Co St Vincent's Primary
55.816555, -3.39989, 7, Paynterhey C Of E Primary School
55.816555, -3.39989, 7, Broadmead Primary School
55.816555, -3.39989, 7, Powerboat & RIB
55.816555, -3.39989, 7, Clyst Hydon Primary School
55.816555, -3.39989, 7, Plymtree Pre-school
55.816555, -3.39989, 7, Plymtree C Of E Primary School
55.816555, -3.39989, 7, Broadmeadbury Church of England Primary School
55.816555, -3.39989, 7, The Forest School at Escot
55.816555, -3.39989, 7, Awliscombe Primary School
55.816555, -3.39989, 7, Accent Studios
55.816555, -3.39989, 7, Kentisbeare C Of E Primary School
55.816555, -3.39989, 7, St. Peter's Belgrave Primary and Pre-school
55.816555, -3.39989, 7, Whimple Preschool Playgroup
55.816555, -3.39989, 7, Whimple Primary School
55.816555, -3.39989, 7, A B C Day Nursery
55.816555, -3.39989, 9, North Walls Primary School
55.816555, -3.39989, 9, North Walls Junior High School
55.816555, -3.39989, 9, Little wards
55.816555, -4.39989, 4, ABCD School
```

## POSTMAN DELETE

<https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda>

The screenshot shows the Postman interface with a DELETE request to the specified endpoint. The request body is a JSON object with a single key 'data' containing an array of objects, each with 'action' set to 'delete' and 'school' set to the coordinates and name of a school. The response status is 200 OK, and the body contains the message '1 School deleted'.

This screenshot shows the same DELETE request in Postman, but with a focus on the 'Tests' tab. It includes three test cases: one for a valid response body, one for processing, and a status test. All tests have passed.

The screenshot shows a GET request to the same endpoint, resulting in a large list of schools. The list includes various names and coordinates, such as 'Bellaheston Academy' at 55.816555, -4.39899, and 'Tinto Primary School' at 55.816555, -4.39899. The response body is a JSON array of approximately 100 school entries.

Home Workspaces API Network

My Workspace

Collections

APIs

Environments

History

REST API basics: CRUD, test & variable

Find Nearby Schools

getShools

getShoolsErr

storeShool

DEL deleteShool

DEL deleteShoolErr1

DEL deleteShoolErr2

DELETE https://qyk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body (x-www-form-urlencoded)

```
1 {  
2   "data": {},  
3   "action": "",  
4   "school": "Glasgow Club Bellahouston"  
5 }  
6  
7 }
```

Body Cookies Headers (10) Test Results (1/1)

All Passed Skipped Failed

PASS Status test

Status: 400 Bad Request Time: 715 ms Size: 486 B Save as example

Home Workspaces API Network

My Workspace

Collections

APIs

Environments

History

REST API basics: CRUD, test & variable

Find Nearby Schools

getShools

getShoolsErr

storeShool

DEL deleteShool

DEL deleteShoolErr1

DEL deleteShoolErr2

DELETE https://qyk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda

Params Authorization Headers (8) Body Pre-request Script Tests Settings

```
1 pm.test("Status test", function () {  
2   pm.response.to.have.status(400);  
3 });
```

Body Cookies Headers (10) Test Results (1/1)

All Passed Skipped Failed

PASS Status test

Status: 400 Bad Request Time: 715 ms Size: 486 B Save as example

Home Workspaces API Network

My Workspace

Collections

APIs

Environments

History

REST API basics: CRUD, test & variable

Find Nearby Schools

getShools

getShoolsErr

storeShool

DEL deleteShool

DEL deleteShoolErr1

DEL deleteShoolErr2

DELETE https://qyk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body (x-www-form-urlencoded)

```
1 {  
2   "data": {},  
3   "action": "delete",  
4   "school": 123  
5 }  
6  
7 }
```

Body Cookies Headers (10) Test Results (1/1)

All Passed Skipped Failed

PASS Status test

Status: 502 Bad Gateway Time: 1405 ms Size: 517 B Save as example

The screenshot shows the Postman interface with the following details:

- Header:** Home, Workspaces, API Network, Search Postman.
- Left Sidebar:** My Workspace, Collections, APIs, Environments, History.
- Collection:** Find Nearby Schools (selected), REST API basics: CRUD, test & variable.
- API Requests:**
  - GET getShools
  - GET getShoolsErr
  - POST storeShool
  - DEL deleteShoolErr2
  - DEL deleteShoolErr1
  - DEL deleteShool
  - REST API basics: CRUD, test & variable
    - GET Get data
    - POST Post data
    - PUT Update data
    - DEL Delete data
- Run Results:** Find Nearby Schools - Run results, Ran today at 18:47:46. Test summary: 12 tests, 12 passed, 0 failed, 0 skipped. Avg. Resp. Time: 1454 ms.
- Test Examples:**
  - GET getShools: PASS response must be valid and have a body, PASS response should be okay to process, PASS Status test.
  - POST storeShool: PASS Status test, PASS response should be okay to process, PASS response must be valid and have a body.
  - DELETE deleteShoolErr2: PASS Status test.
  - DELETE deleteShoolErr1: PASS Status test.
  - DELETE deleteShool: PASS response must be valid and have a body, PASS response should be okay to process, PASS Status test.
- Bottom:** Online, Console, Postbot, Runner, Auto-select agent, Cookies, Trash.

The screenshot shows the Postman interface with the following details:

- Header:** Home, Workspaces, API Network, Search Postman.
- Left Sidebar:** My Workspace, Collections, APIs, Environments, History.
- Collection:** Find Nearby Schools (selected), REST API basics: CRUD, test & variable.
- API Requests:** Same as the first screenshot.
- Run Results:** Find Nearby Schools - Run results, Ran today at 18:47:46. Test summary: 12 tests, 12 passed, 0 failed, 0 skipped. Avg. Resp. Time: 1454 ms.
- Test Results Summary:** A grid showing the count of passes and fails for each request type:
 

Request Type	Passes	Fails
GET getShools	3	0
GET getShoolsErr	1	0
POST storeShool	3	0
DEL deleteShoolErr2	1	0
DEL deleteShoolErr1	1	0
DEL deleteShool	3	0
- Bottom:** Online, Console, Postbot, Runner, Auto-select agent, Cookies, Trash.

The screenshot shows the Postman interface with the following details:

- Header:** Home, Workspaces, API Network, Search Postman.
- Left Sidebar:** My Workspace, Collections, APIs, Environments, History.
- Collection:** Find Nearby Schools (selected), REST API basics: CRUD, test & variable.
- API Requests:** Same as the previous screenshots.
- Test Results:** A detailed view of the test results for each request:
  - GET getShools: Tests: response must be valid and have a body, response should be okay to process, Status test.
  - GET getShoolsErr: Tests: Status test.
  - POST storeShool: Tests: Status test, response should be okay to process, response must be valid and have a body.
  - DEL deleteShoolErr2: Tests: Status test.
  - DEL deleteShoolErr1: Tests: Status test.
  - DEL deleteShool: Tests: response must be valid and have a body, response should be okay to process, Status test.
- Bottom:** Online, Console, Postbot, Runner, Auto-select agent, Cookies, Trash.

## POSTMAN Documentation

<https://web.postman.co/workspace/dc3d0e94-a778-4f62-8070-d365d6b604da/documentation/7516849-c753e56f-17c1-41ae-8154-3b8104092425>

### **Find Nearby Schools**

<https://documenter.getpostman.com/view/7516849/2sA2xe3ZDz>

### **6. Review your code:**

- Look over all parts of the code carefully. Make sure it's secure and works as expected. Document any changes you make, meaning write down what you've modified or fixed.

## **TECHNICAL REQUIREMENTS:**

### **User function - in Google Colab**

Function run in a Google Colab

API keys and any other sensitive data must NEVER be exposed (at any point during the design, development, testing or deployment phases)

### **API function - in AWS Lambda**

Must be secured

Will receive data

Will accept a list of schools and add them to a database or file

Will accept a name of a school to delete from the file on S3

Will return a list of all schools in the database or file on request  
Will return a success status and message after adding or deleting.  
Will return an error status and message on encountering a problem  
Will always return success or error

## SUBMISSION

### 1. Google Colab Notebook:

- Create a Google Colab notebook that includes all the function code for finding schools and sending requests to the API.

<https://colab.research.google.com/drive/1ceFgMsmRSHfo8LWYNTMVrjWxK2yu4bY1#scrollTo=H5JifWHib4q8>

### 2. Test Plan Document:

- Write a document (like a Word file) that explains your test plan. This document should detail how you'll test the system.

**Test Plan Document.odt**

### 3. Testing Evidence in Google Colab:

- Show evidence that you've tested the function code in Google Colab. This could be screenshots or descriptions of your testing process.

<https://colab.research.google.com/drive/1ceFgMsmRSHfo8LWYNTMVrjWxK2yu4bY1#scrollTo=H5JifWHib4q8>

#### 4. **Zipped Folder for AWS Lambda:**

- Zip the folder that contains the code you uploaded to AWS Lambda. This is the code that runs your functions in the cloud.

**function.zip**

#### 5. **Lambda Function Testing Screenshots:**

- Take screenshots of your Lambda function being tested. This shows that it works as expected in the AWS environment.

**Lambda Function Testing Screenshots.zip**

#### 6. **IAM Policies Screenshot:**

- Take a screenshot of the IAM (Identity and Access Management) policies related to your Lambda function. This shows how permissions are set up.

**IAM Policies Screenshot.zip**

#### 7. **API Documentation:**

**API Documentation.odt**

<https://web.postman.co/workspace/dc3d0e94-a778-4f62-8070-d365d6b604da/documentation/7516849-c753e56f-17c1-41ae-8154-3b8104092425>

<https://documenter.getpostman.com/view/7516849/2sA2xe3ZDz>

#### 8. **Postman Test Screenshots:**

- Take screenshots of all the tests you've run in Postman. This shows that your API works properly and responds correctly to different requests.

**Postman Test Screenshots.zip**