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 and 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.

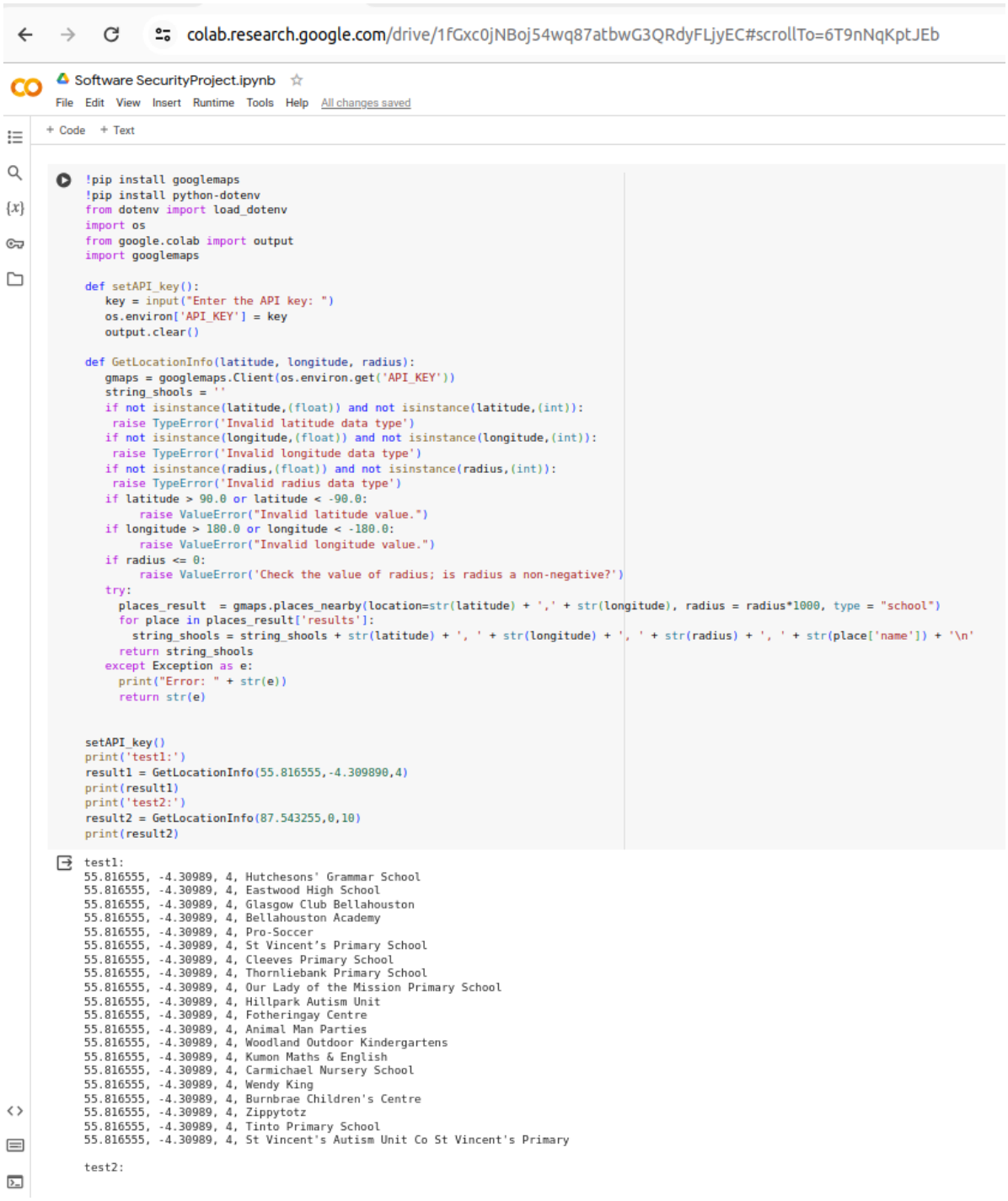
Colab/local functions:

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.



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.

```python
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: lVjE3I0uUdaU72VgUHws45jeOsEeEUurhpWKTSi9
<Response [200]>
```

```python
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: lVjE3I0uUdaU72VgUHws45jeOsEeEUurhpWKTSi9
'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, Bellahousto
n 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, -4.3
0989, 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\n5
5.816555, -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, -4.309
89, 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

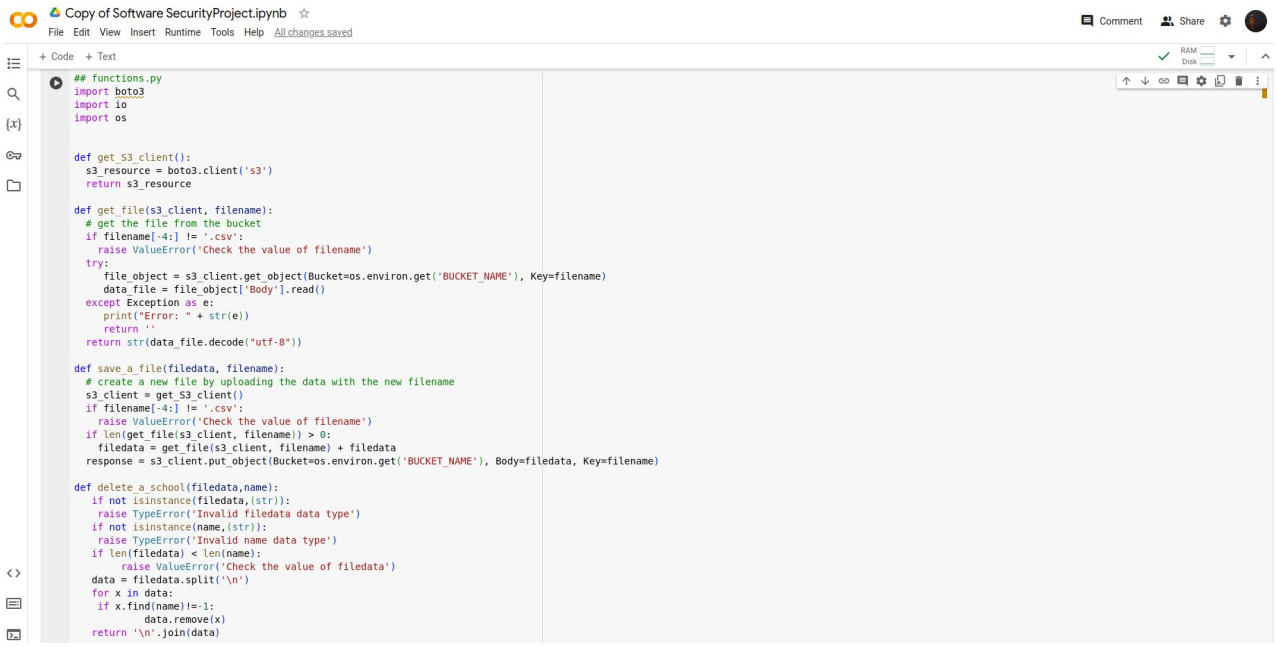
- make sure you have a function set up on AWS Lambda to receive the request to either store a new set of schools or to get the whole list from the file and return it

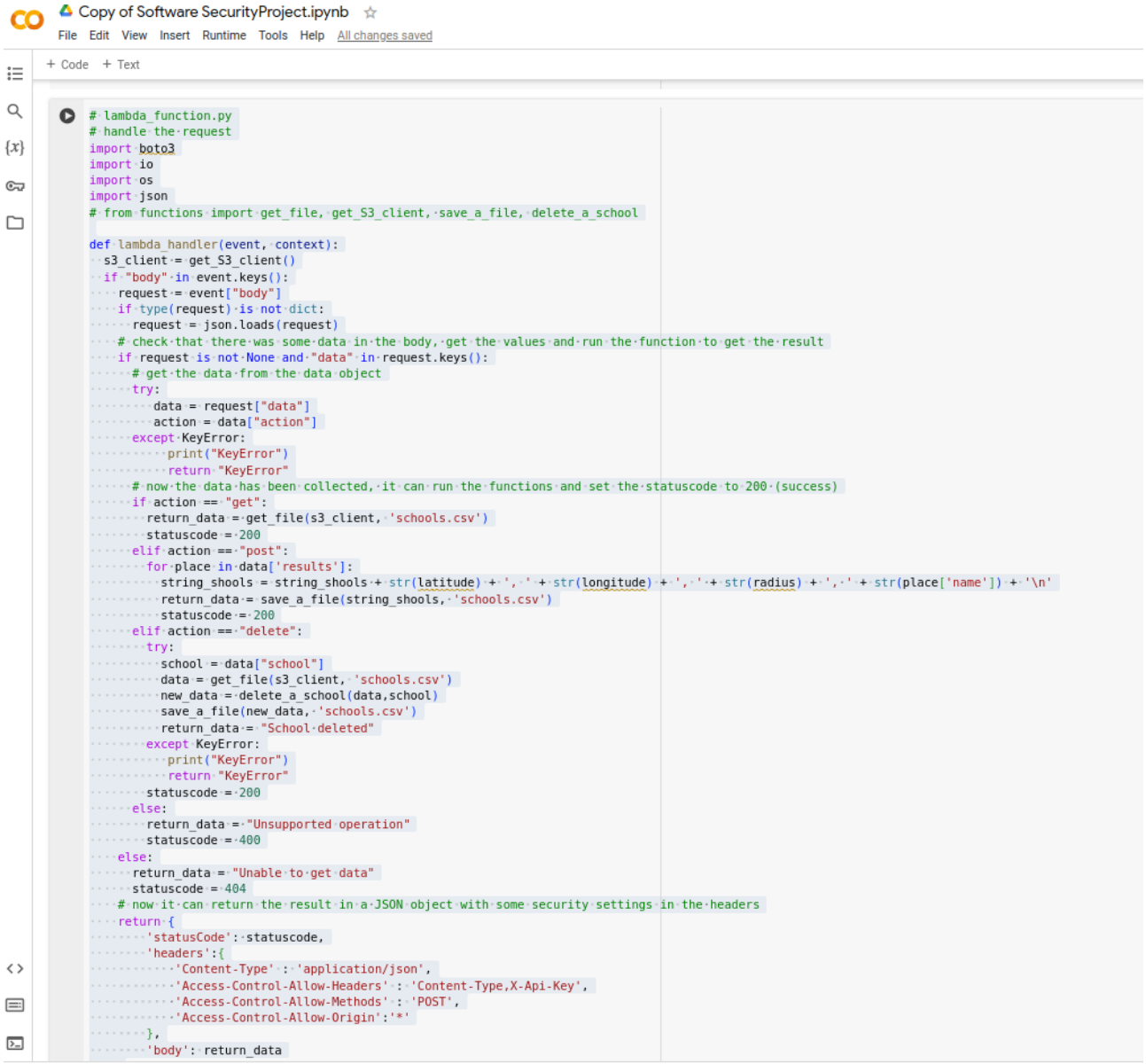
Format the list of schools ready to add to a CSV (Comma-Separated Values) text file.

- 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

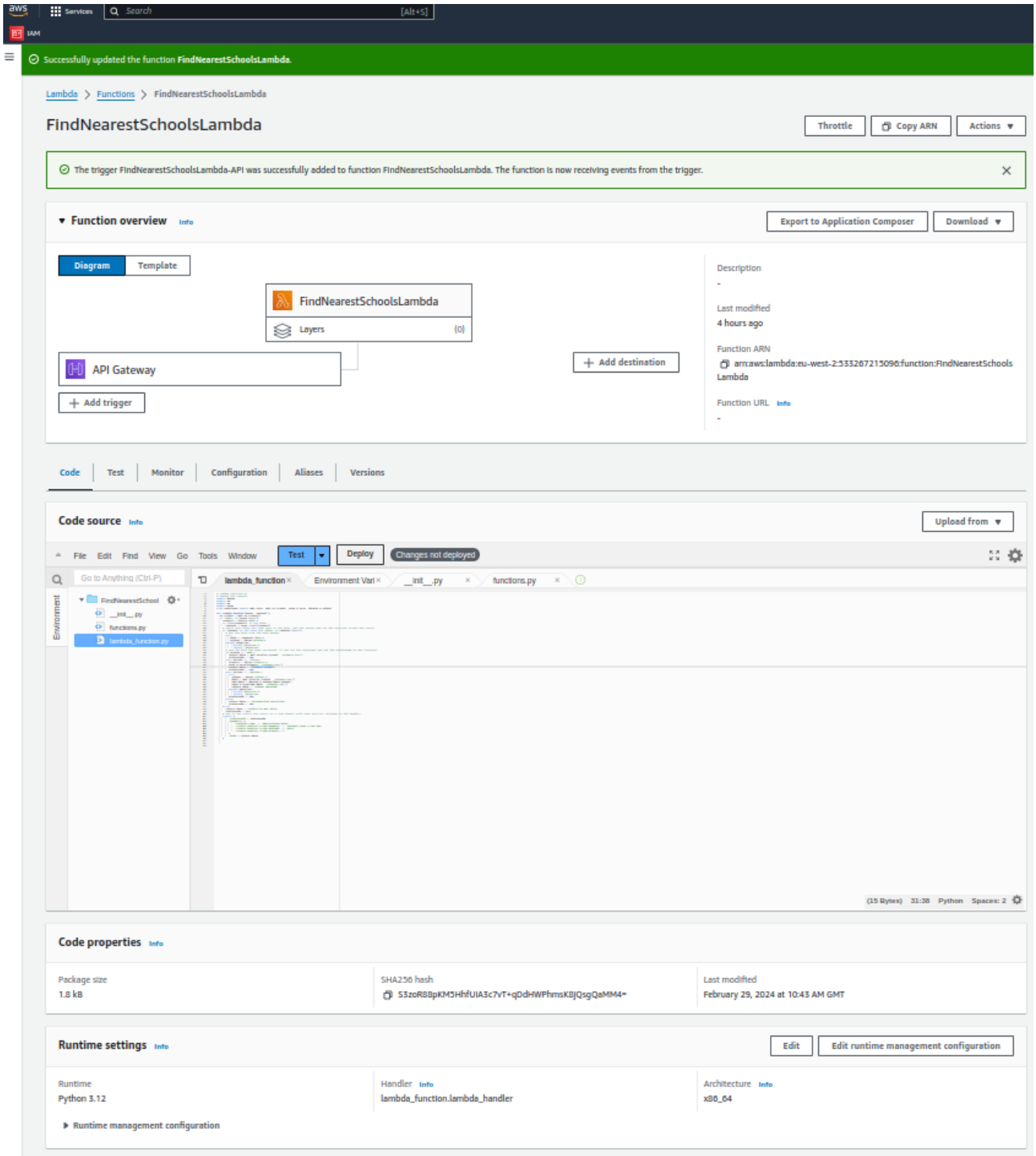

Copy of Software SecurityProject.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
functions.py
import boto3
import io
import os
def get_S3_client():
s3_resource = boto3.client('s3')
return s3_resource
def get_file(s3_client, filename):
# get the file from the 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 ''
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
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
# 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

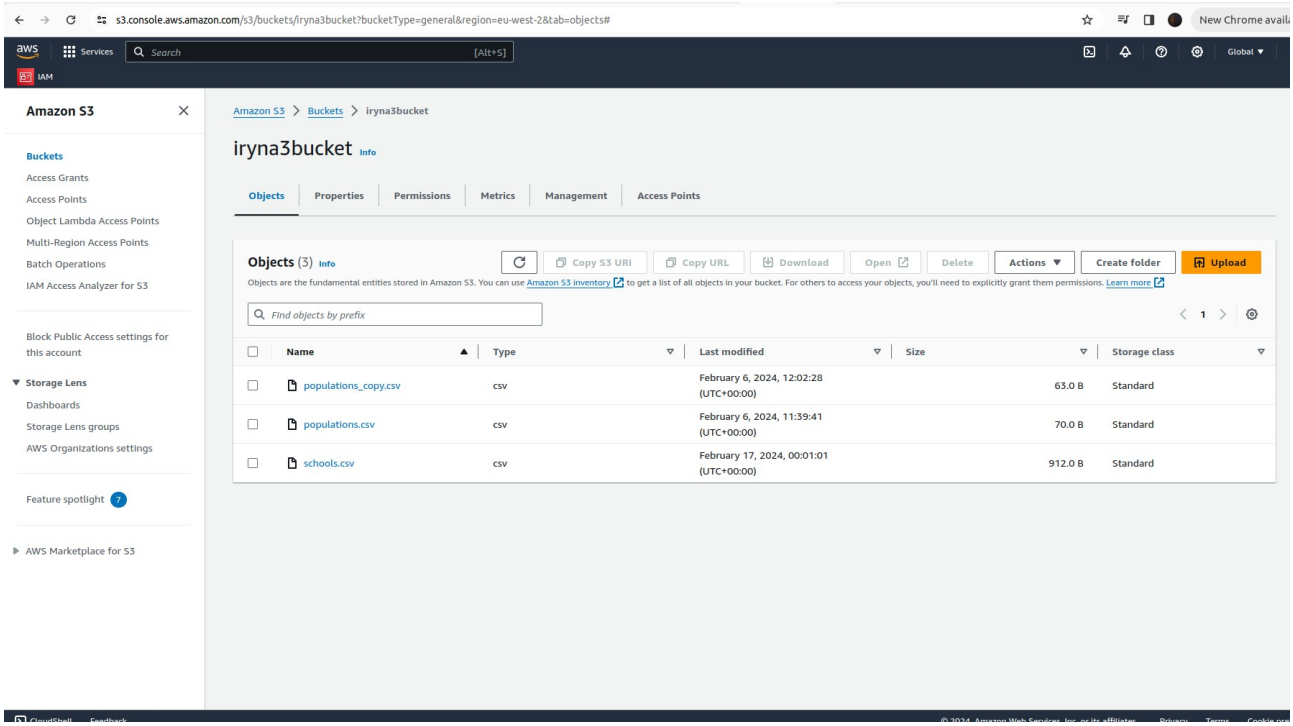
1. **Create Serverless Function**:

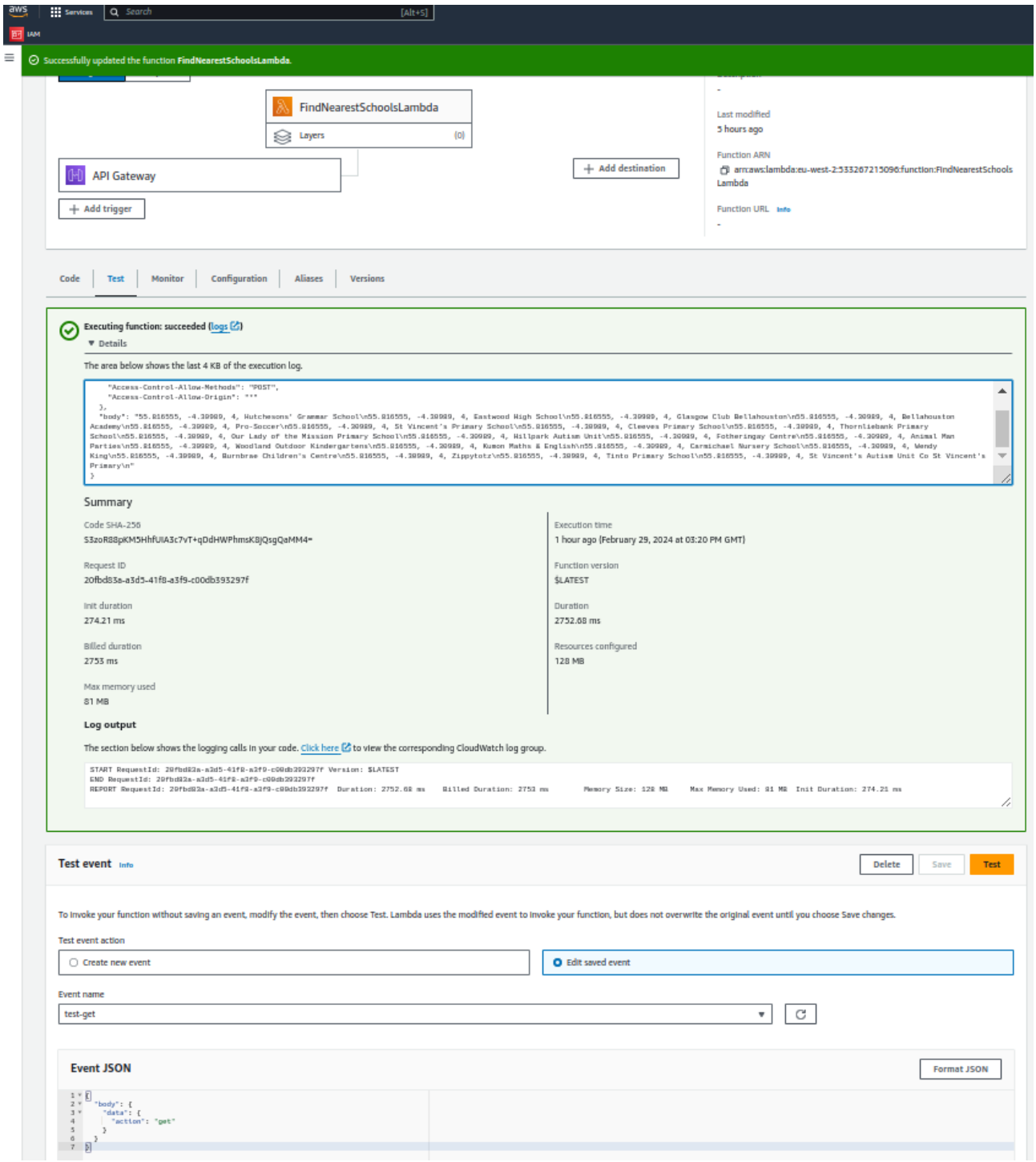
- Use AWS Lambda, a service provided by Amazon, to build a function that doesn't need a server to run.



2. **Access S3:**

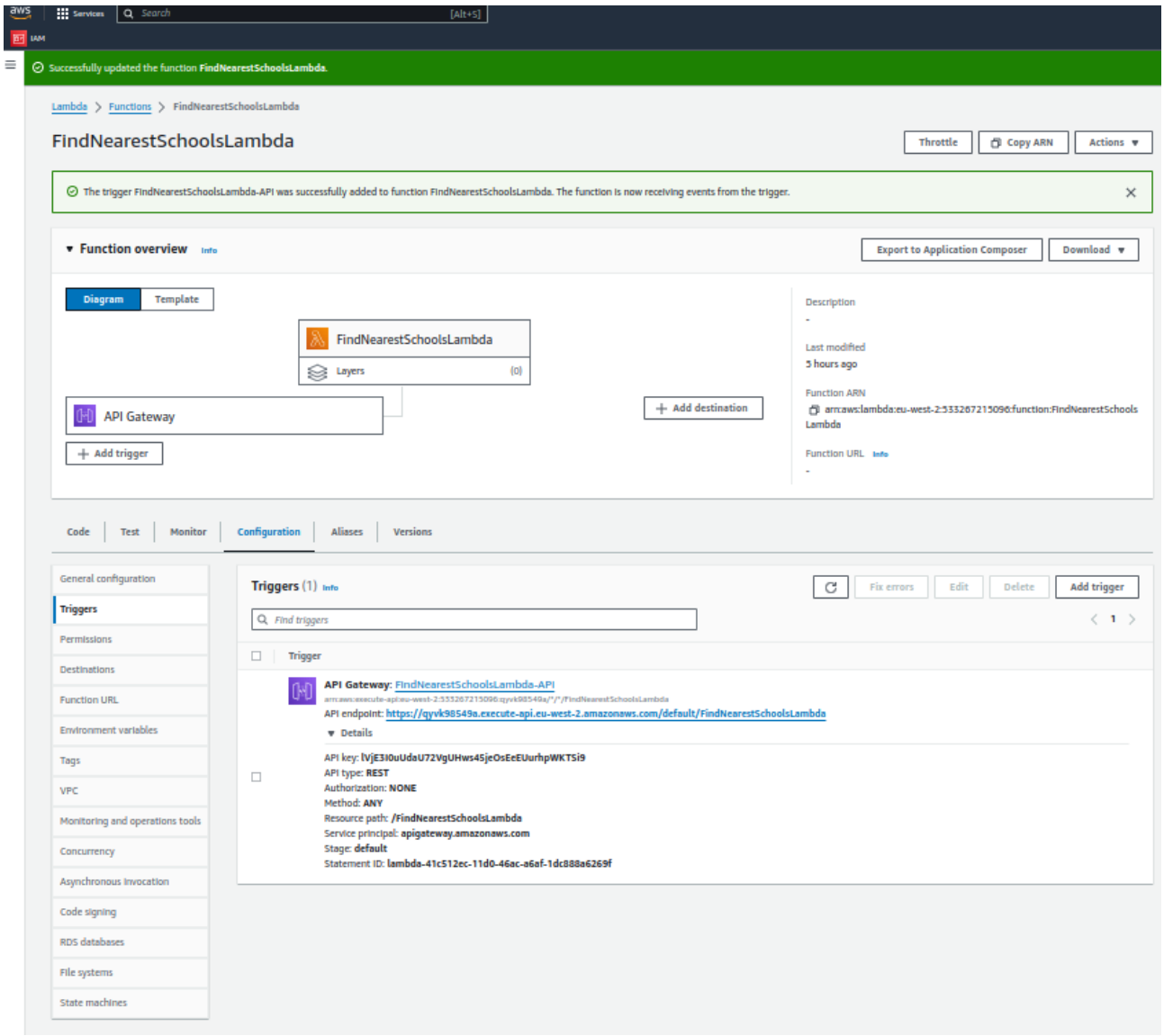
- Make sure your Lambda function can read and write files in Amazon S3, a storage service provided by Amazon.





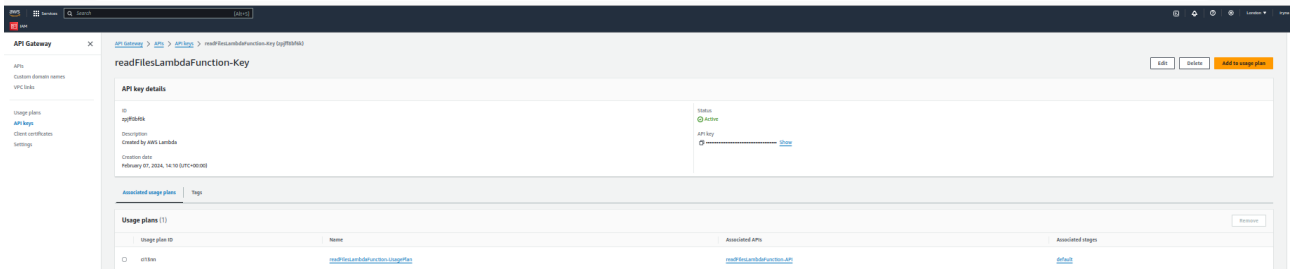
3. **Trigger with API Gateway**:

- 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.



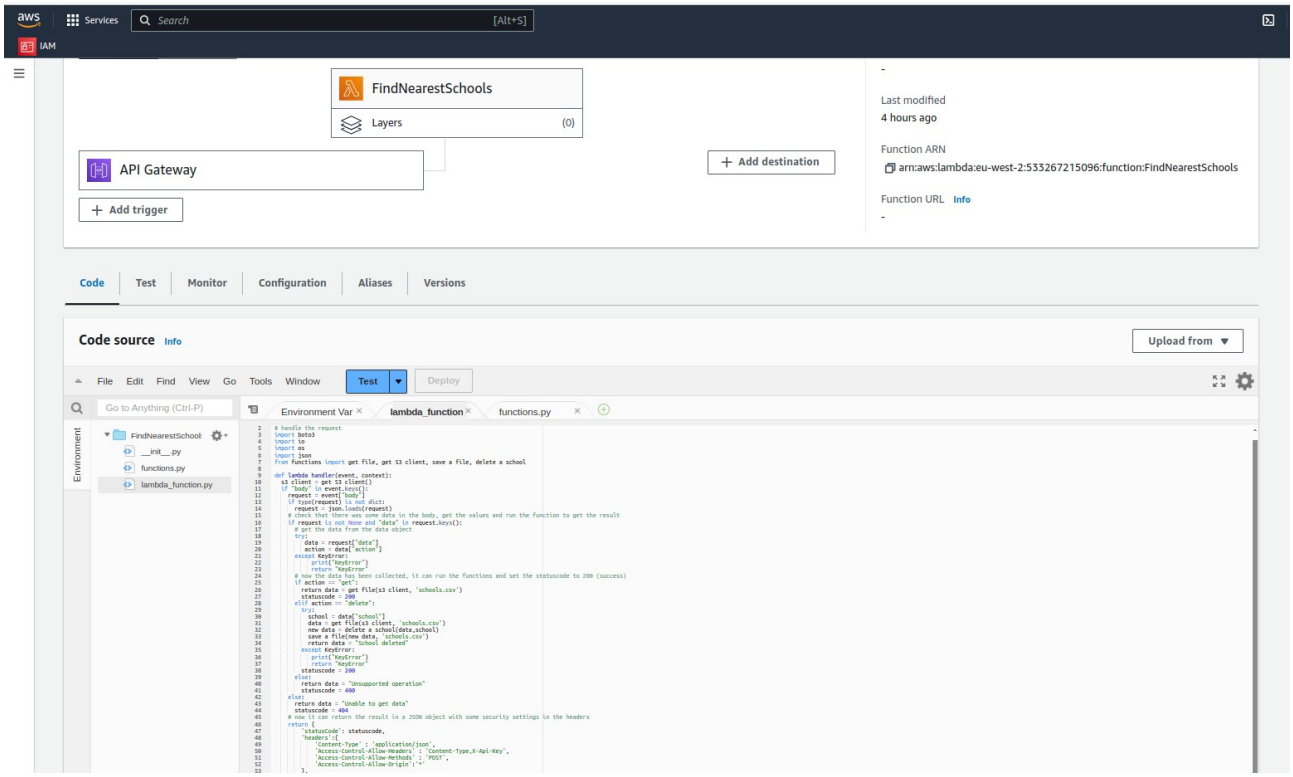
4. **Secure with API Key**:

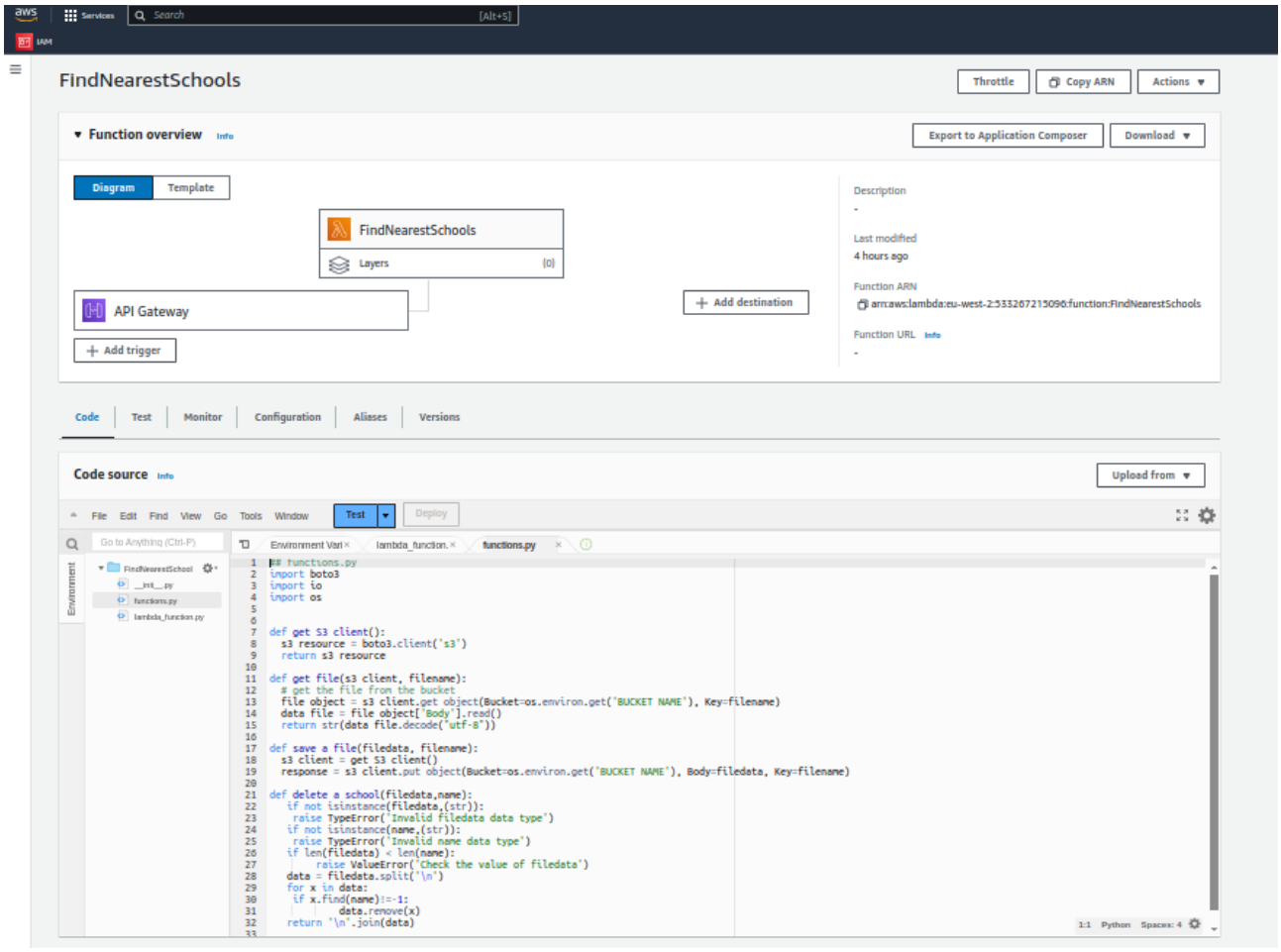
- Make sure your API Gateway is protected by an API key. This means people need a special key to access your API and trigger your Lambda function.



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.


aws
Services
Search
[Alt+S]
IAM
FindNearestSchools
Layers
(0)
API Gateway
Add trigger
Add destination
Last modified
4 hours ago
Function ARN
arn:aws:lambda:eu-west-2:533267215096:function:FindNearestSchools
Function URL
Info
Code
Test
Monitor
Configuration
Aliases
Versions
Code source
Info
Upload from
File
Edit
Find
View
Go
Tools
Window
Test
Deploy
Environment Var
lambda_function
functions.py
Environment
FindNearestSchool
__init__.py
functions.py
lambda_function.py


aws
Services
Search
[Alt+S]
IAM
FindNearestSchools
Throttle
Copy ARN
Actions
Function overview
Info
Export to Application Composer
Download
Diagram
Template
FindNearestSchools
Layers
(0)
API Gateway
Add trigger
Add destination
Description
-
Last modified
4 hours ago
Function ARN
arn:aws:lambda:eu-west-2:533267215096:function:FindNearestSchools
Function URL
Info
Code
Test
Monitor
Configuration
Aliases
Versions
Code source
Info
Upload from
File
Edit
Find
View
Go
Tools
Window
Test
Deploy
Environment Var
lambda_function
functions.py
Environment
FindNearestSchool
__init__.py
functions.py
lambda_function.py
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
file_object = s3_client.get_object(Bucket=os.environ.get('BUCKET NAME'), Key=filename)
data_file = file_object['Body'].read()
return str(data_file.decode('utf-8'))
def save_a_file(filedata, filename):
s3_client = get_S3_client()
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)
1:1
Python
Spaces: 4

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

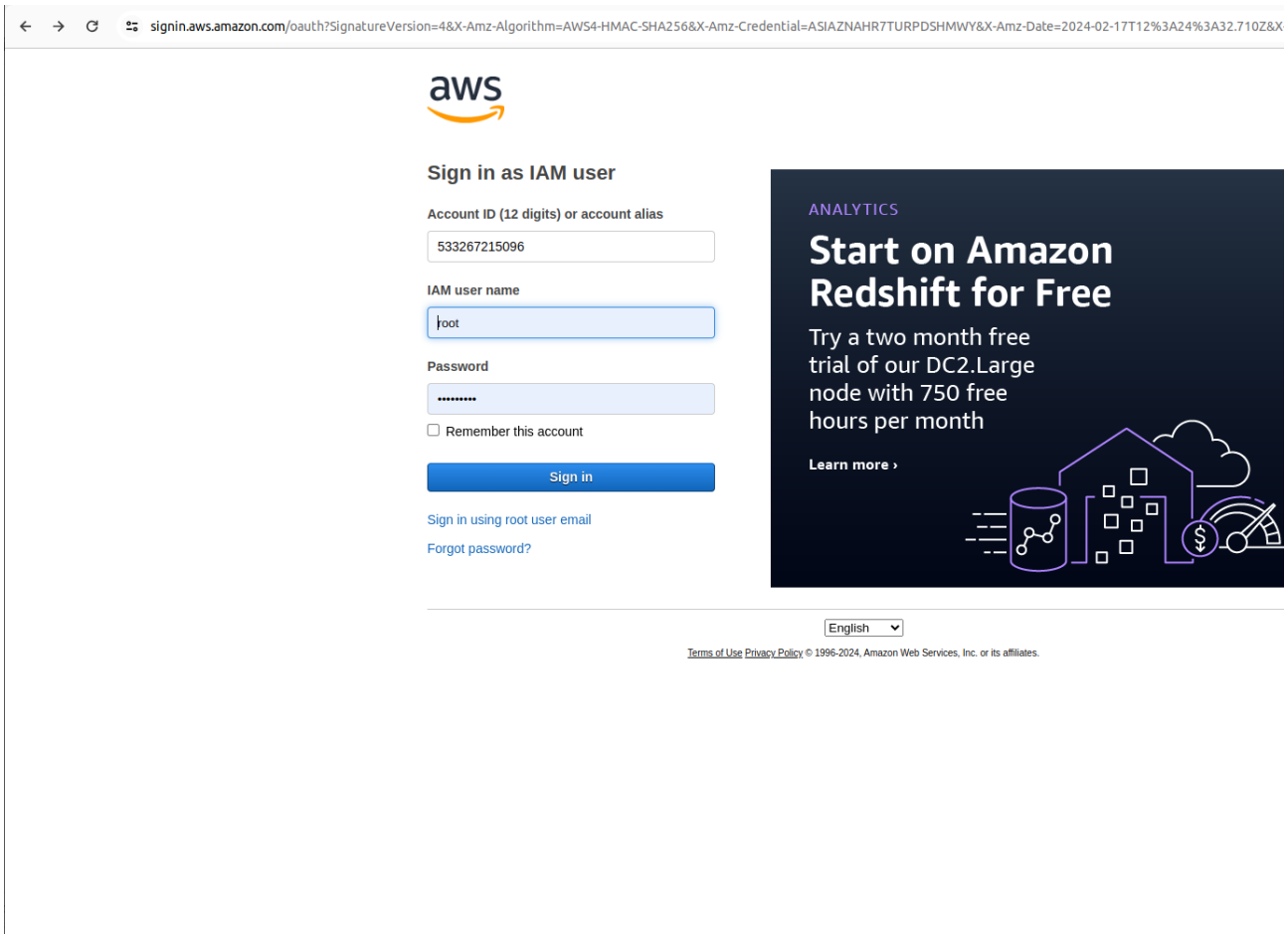
1. **Test for Security Issues**:

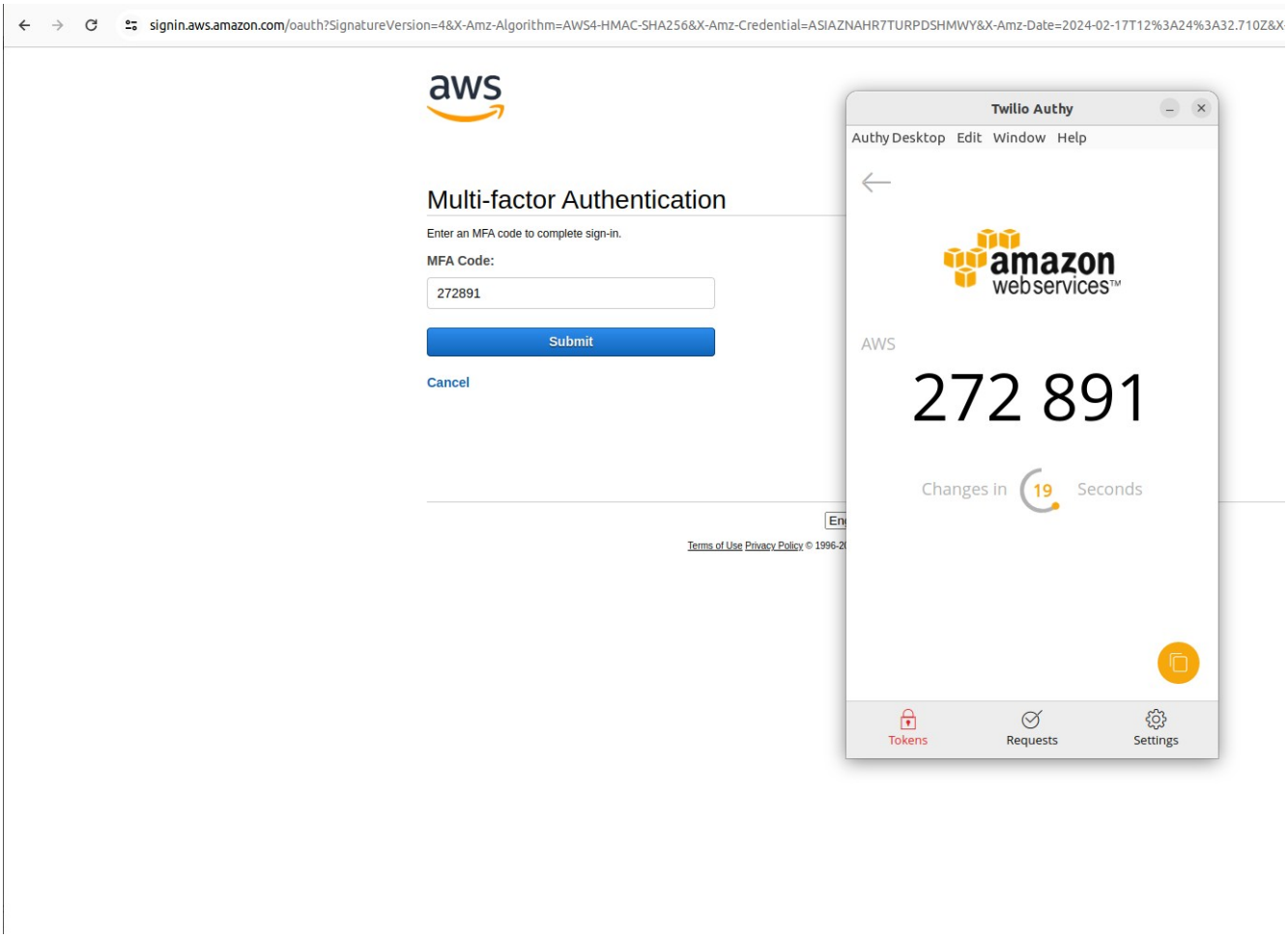
- Check the software to find any security problems. This means looking for ways that unauthorised people could access or mess with the system.

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.

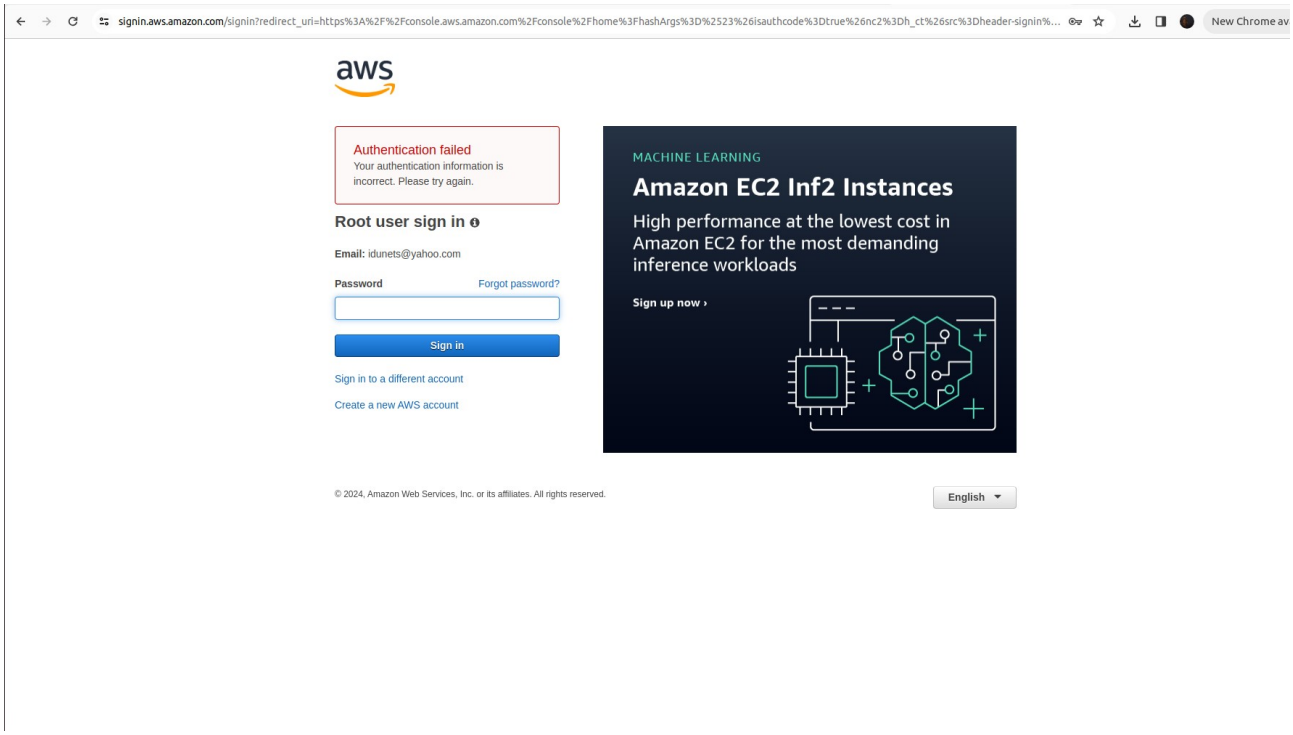
MFA added extra security because it requires users to provide unique authentication from an AWS supported MFA mechanism in addition to their regular sign-in credentials when they access AWS websites or services.

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.

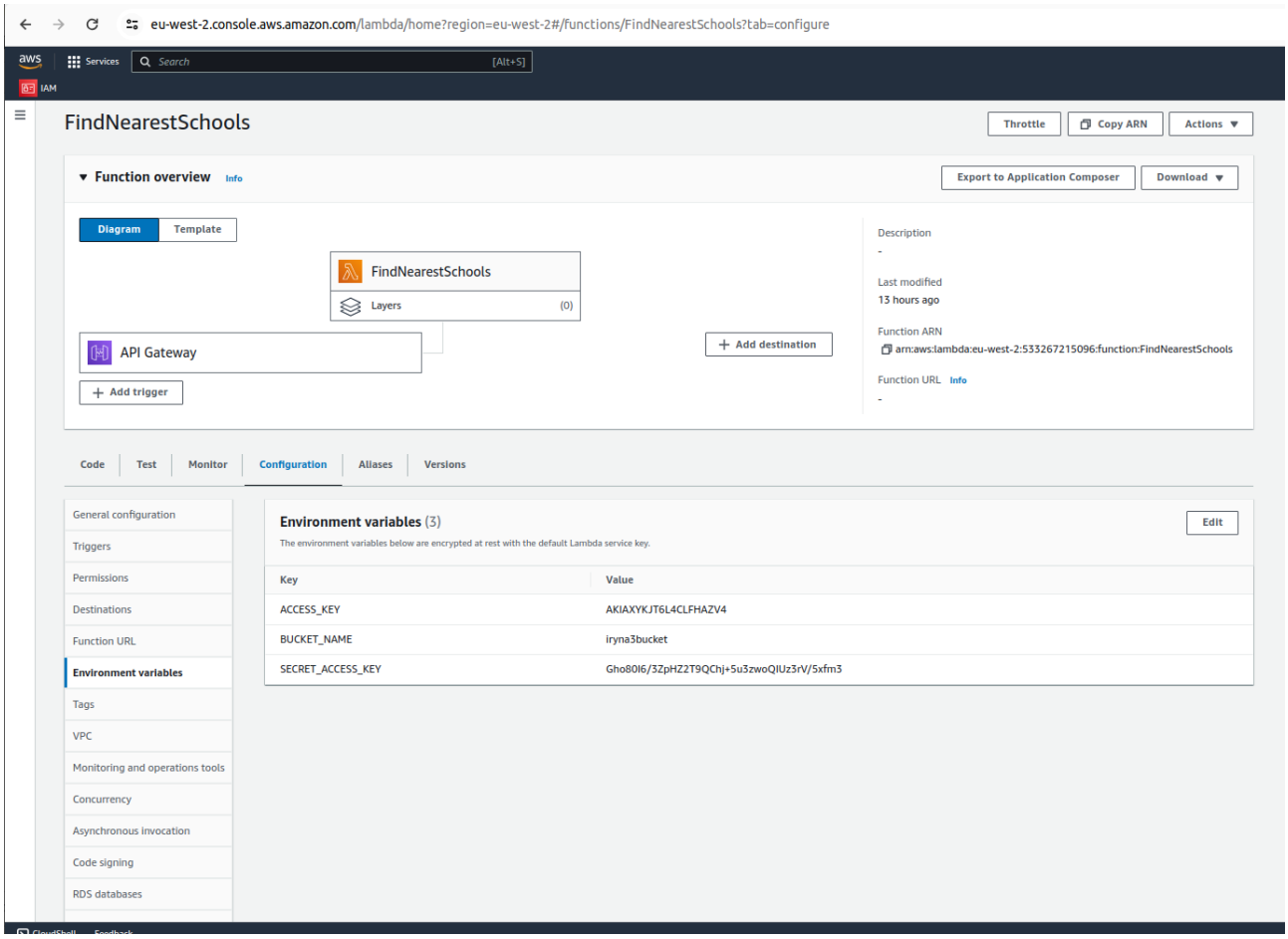




When unauthorised people input wrong login, password or the time-based one-time password (TOTP) they couldn't access AWS system.



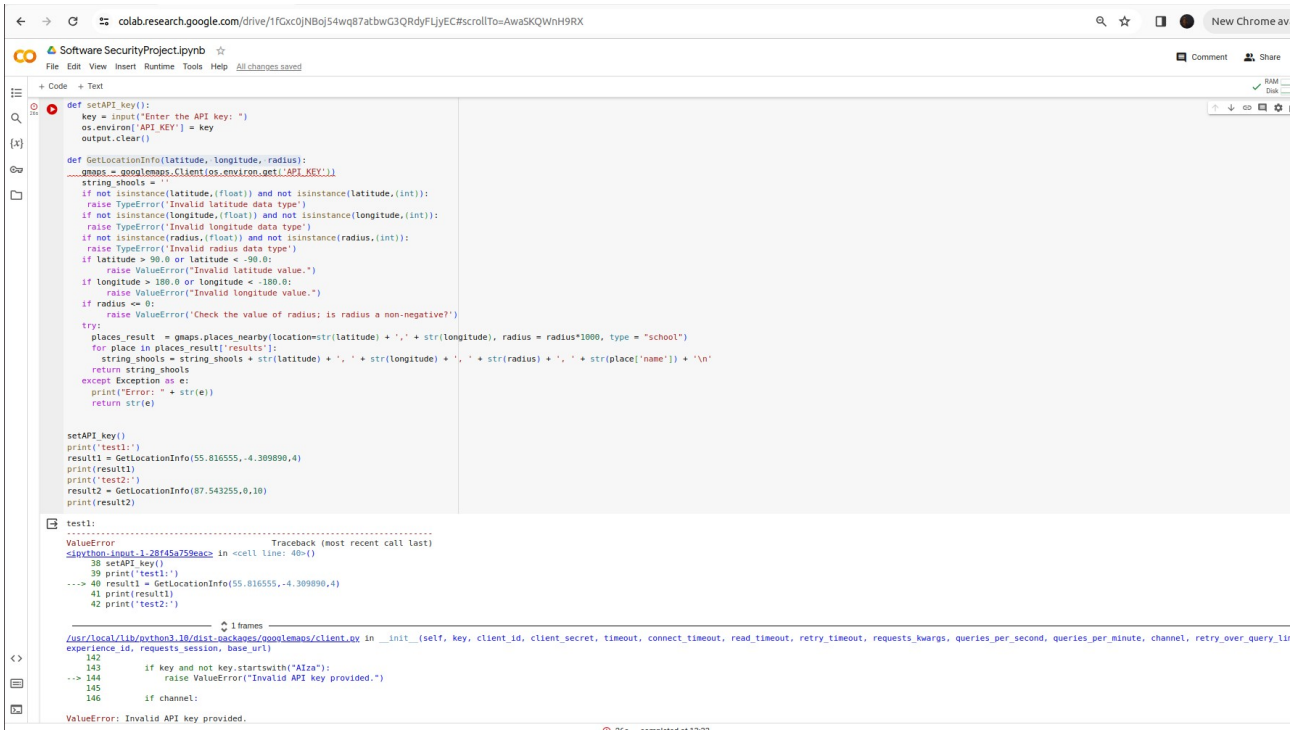
Test2. I kept the ACCESS_KEY, SECRET_ACCESS_KEY and BUCKET_NAME secret, save it as an environment variable:

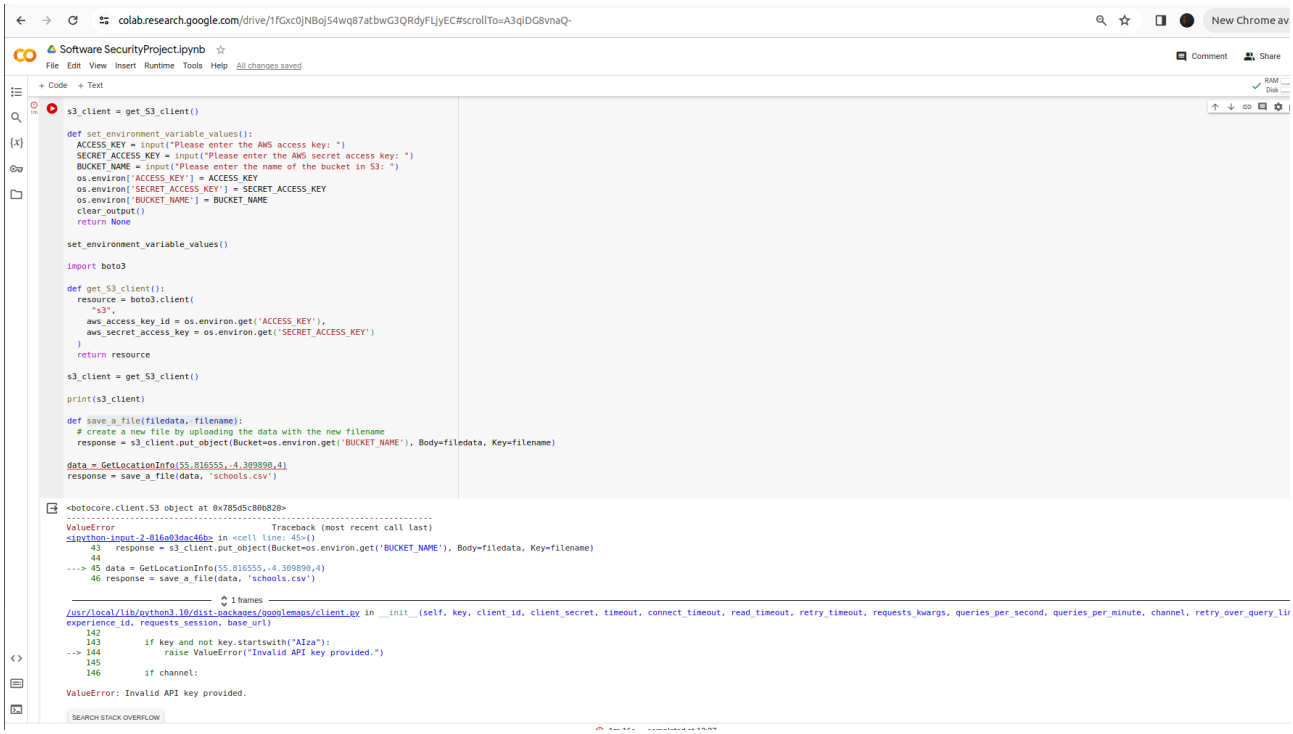


All environment variables that are encrypted to prevent rogue access.

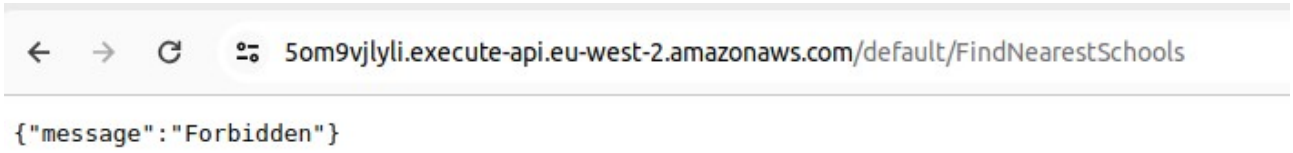
Unauthorized users can't use functions: GetLocationInfo(latitude, longitude, radius) and save_a_file(filedata, filename) without gain access to sensitive information stored in the solution, such as API keys and bucket name. In this case we got :

ValueError: Invalid API key provided.





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.



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 reoeatedly as you develop it).

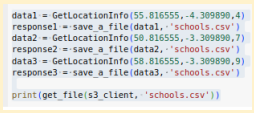
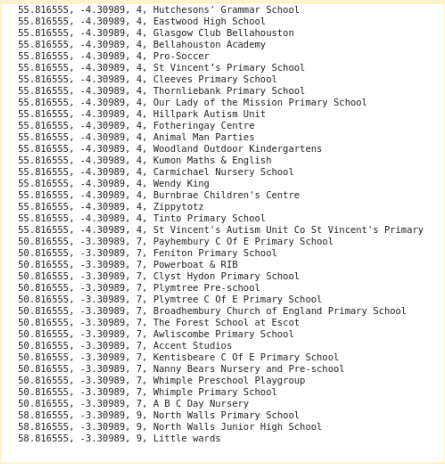
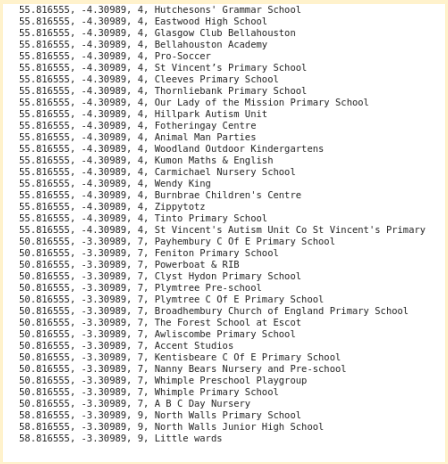
- Create automated tests in your colab

- 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.

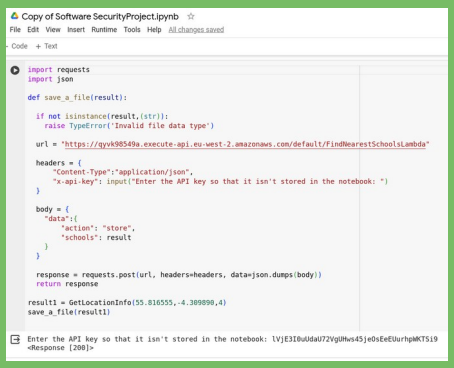
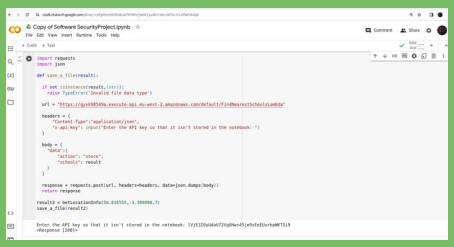
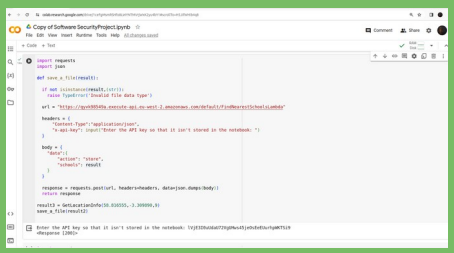
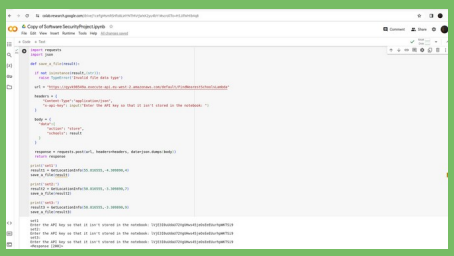
Test no.	What is being tested?	Data (latitude, longitude, radius)	Expected outcome	Actual outcome	Notes
COLAB TESTS					
1.	Test GetLocationInfo(latitude, longitude, radius) With valid data	GetLocationInfo(55.816555,-4.309890,4)	55.816555, -4.30989, 4, Hutchesons' Grammar School 55.816555, -4.30989, 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, 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, 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 Primar	This test works

2.	Test GetLocationInfo(latitude, longitude, radius) With valid data				This test works
			COLAB UNITTESTS		
3.	Test GetLocationInfo(latitude, longitude, radius) With valid data	(55.816555,-4.309890,4) GetLocationInfo(55.816555,-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',GetLocationInfo(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

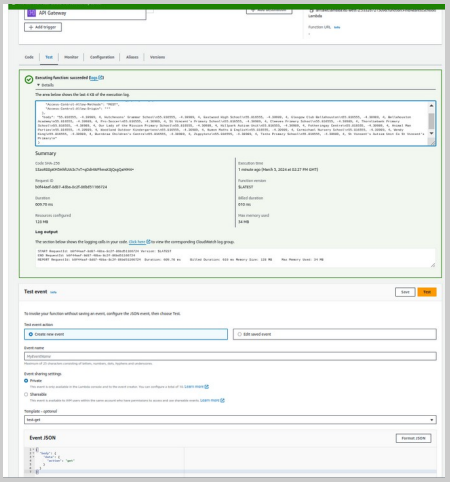
	type				
7.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect longitude data type	55.816555,'abc',4	TypeError	TypeError	This test works
8.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect longitude data type	55.816555,'False', 4	TypeError	TypeError	This test works
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

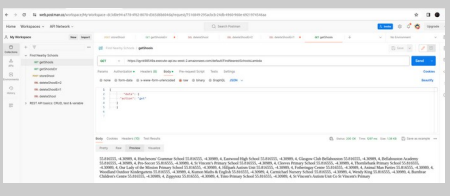
	value				
12.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect longitude data value	-79.999995,-189.230,5	ValueError	ValueError	This test works
13.	Test GetLocationInfo(latitude, longitude, radius) with an incorrect latitude data value	99.999995,0,5	ValueError	ValueError	This test 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 work
21 b.	Test save_a_file(file data) with an correct data	save_a_file(result1).text	"Schools saved"	"Schools saved"	This test work
22.	Test get_file(s3_clie	s3_client,'abc'	ValueError	ValueError	This test

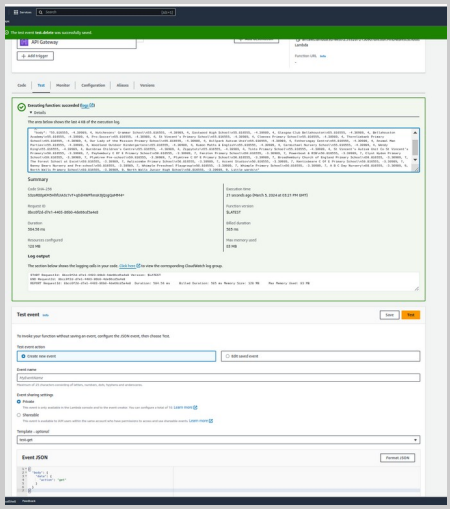
	nt, filename) with an incorrect filename value				works
23.	Test get_file() with an correct data in Colab	get_a_file()	<Response [200]>	 <Response [200]>	This test works
23a	Test get_file() with an correct data in Colab 2 sets	get_a_file()	<Response [200]>	 <Response [200]>	This test works
23b	Test get_file() with an correct data in Colab 3 sets	get_a_file()	<Response [200]>	 <Response [200]>	This test works
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 works
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 works

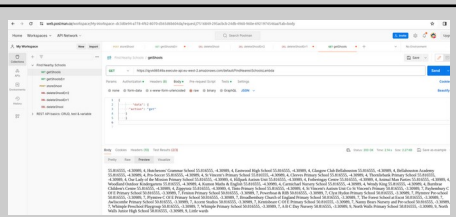
AWS AND POSTMAN TESTS					

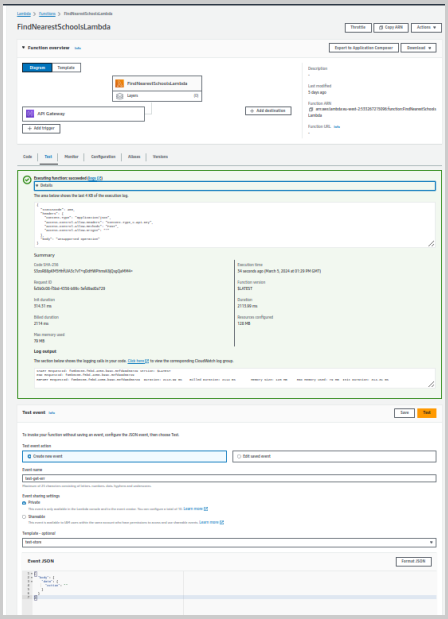
26.	Test AWS Lambda return a list of all schools in the database or file on request with valid key	{
		"body": {
		"data": {
		"action": "get"
		}
		}
		}
			"55.816555, -4.30989, 4, Hutchesons' Grammar School\n55.816555, -4.30989, 4, Eastwood High School\n55.816555, -4.30989, 4, Glasgow Club Bellahouston\n55.816555, -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 & 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"		This test works

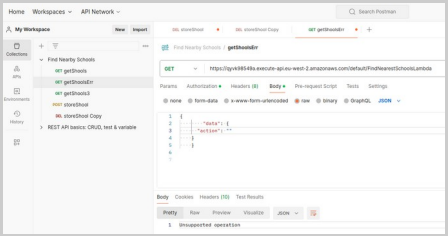
	POSTMAN TESTS	{
		"body": {
		"data": {
		"action": "get"
		}
		}
			55.816555, -4.30989, 4, Hutchesons' Grammar School 55.816555, -4.30989, 4, Eastwood High School 55.816555, -4.30989, 4,		

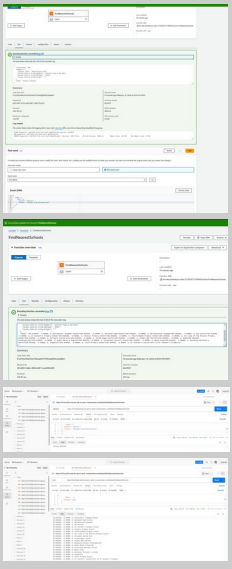
		}	Glasgow Club Bellahouston 55.816555, -4.30989, 4, Bellahouston Academy 55.816555, -4.30989, 4, Pro-Soccer 55.816555, -4.30989, 4, St Vincent’s Primary School 55.816555, -4.30989, 4, Cleeves Primary School 55.816555, -4.30989, 4, Thornliebank Primary School 55.816555, -4.30989, 4, Our Lady of the Mission 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, Zippytotz 55.816555, -4.30989, 4, Tinto Primary School 55.816555, -4.30989, 4, St Vincent's Autism Unit Co St Vincent's Primary		

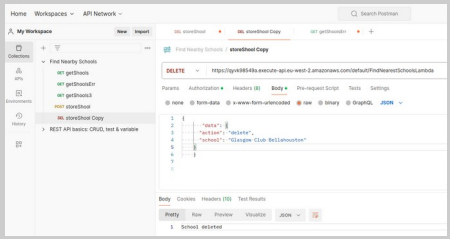
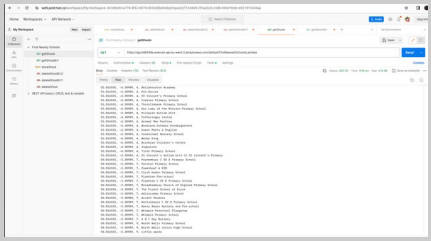
27
a.
	Test AWS Lambda return a list of all schools (three list of data) in the database or file on request with valid key	{
		"body": {
		"data": {
		"action": "get"
		}
		}
		}
			"58.816555, -3.30989, 9, North Walls Primary School\ n58.816555, -3.30989, 9, North Walls Junior High School\ n58.816555, -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 & 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"		This test works

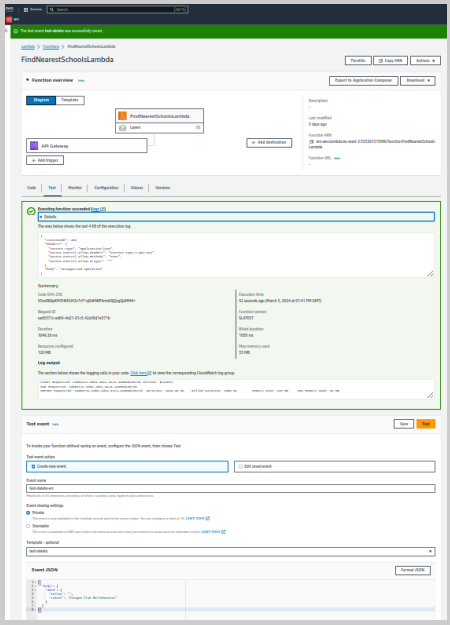
	POSTMAN TESTS	{
		"body": {
		"data": {
		"action": "get"
		}
		}
		}
					This test works

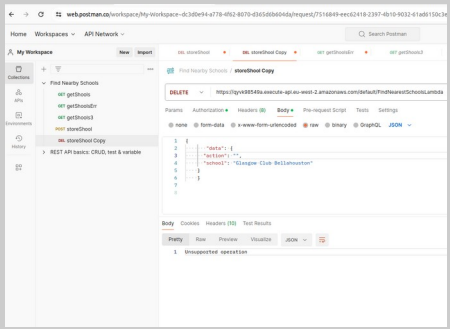
28.	Test AWS Lambda return a list of all schools in the database or file on request with invalid key	{
		"body": {
		"data": {
		"action": ""
		}
		}
		}
			"Unsupported operation"	"Unsupported operation"
				 
					This test works

	POSTMAN TESTS	{
		"body": {
		"data": {
		"action": ""
		}
		}
		}
			"Unsupported operation"		This test works

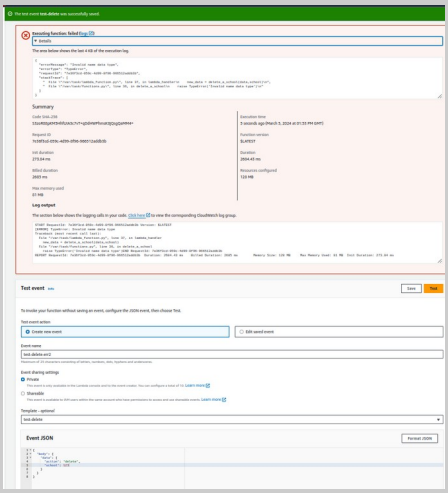
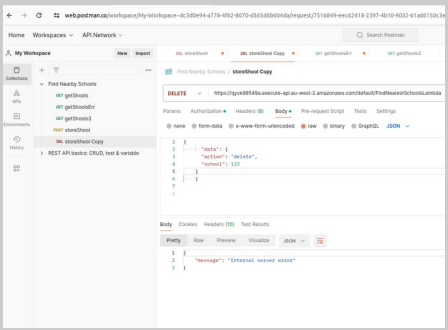
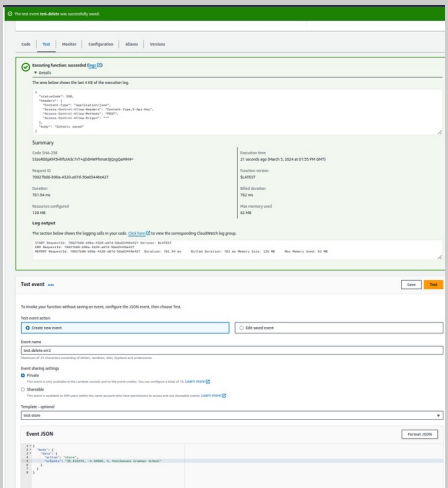
29.	Test AWS Lambda will accept a name of a school to delete from the file on S3 with ivalid key	{
		"body": {
		"data": {
		"action": "delete",
		"school": "Glasgow Club Bellahouston"
		}
		}
		}
			"School deleted"	"School deleted"
				 
					This test works

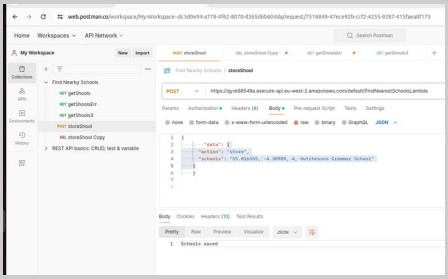
	POSTMAN TESTS	{
		"body": {
		"data": {
		"action": "delete",
		"school": "Glasgow Club Bellahouston"
		}
		}
		}
			"School deleted"	
				 
					This test works

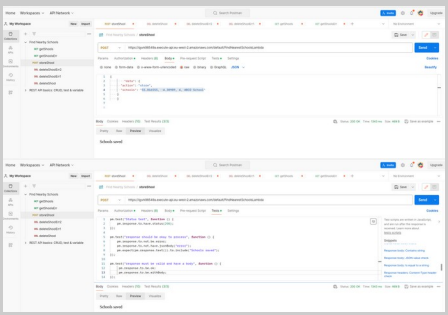
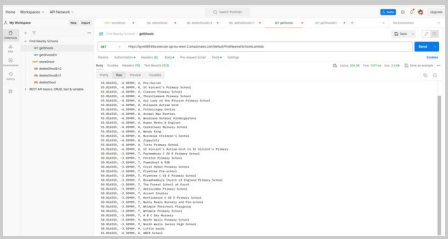
30.	Test AWS Lambda will accept a name of a school to delete from the file on S3 with ivalid key	{
		"body": {
		"data": {
		"action": "",
		"school": "Glasgow Club Bellahouston"
		}
		}
		}
			"Unsupported operation"	"Unsupported operation"
				 
					This test works

	POSTMAN TESTS	{
		"body": {
		"data": {
		"action": "",
		"school": "Glasgow Club Bellahouston"
		}
		}
		}
			"Unsupported operation"		

31.	Test AWS Lambda will accept a	{ "body": {	"errorMessage": "Invalid name data type",	"errorMessage": "Invalid name data type",	This test show

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

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

	POSTMAN TESTS	{
		"body": {
		"data": {
		"action": "store",
		"schools": "55.816555, -4.30989, 4, ABCD School"
		}
		}
		}
			"Schools saved"	
				 
					This test works

4. **Test in Steps**:

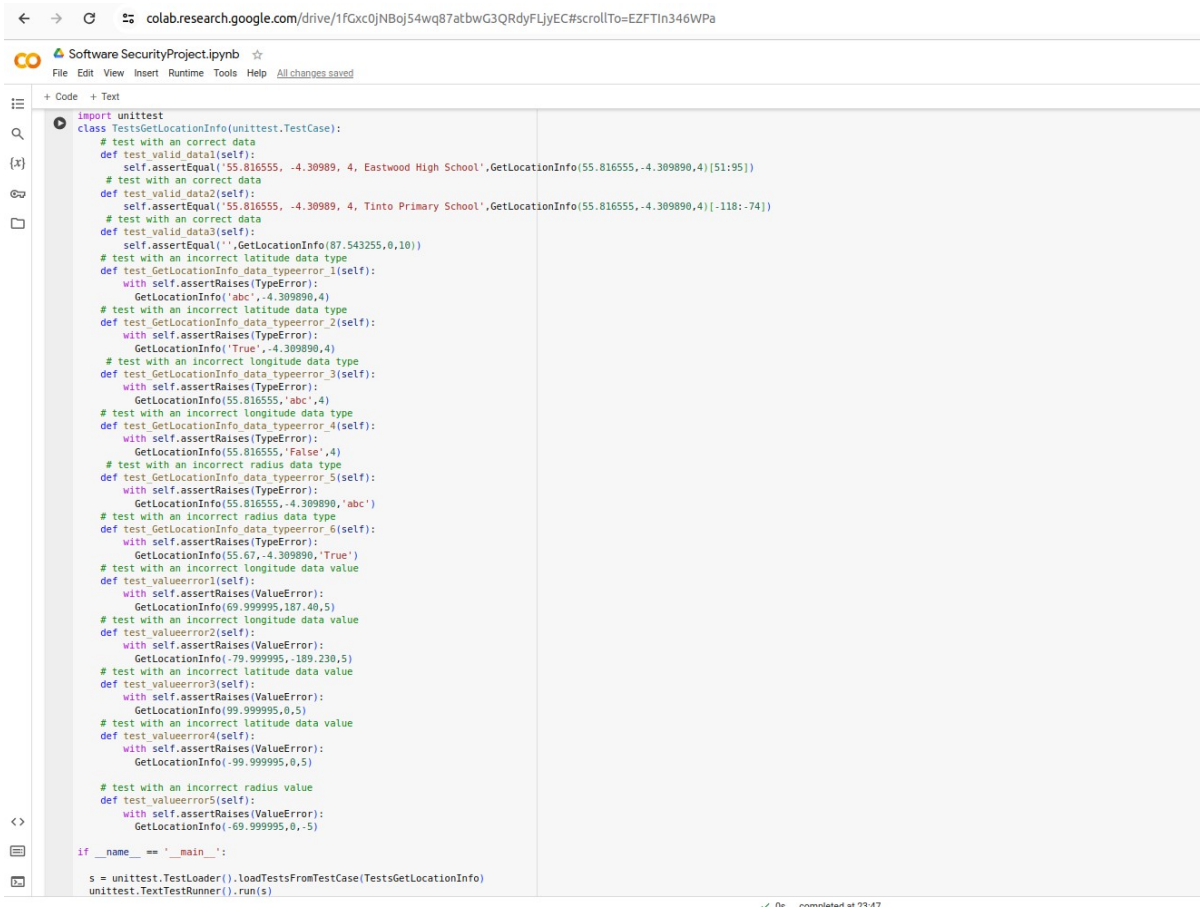
- Start by testing the code on your own computer without sending data to the API. Then, test it again but this time, actually sending data.

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 latitude data type
    def test_GetLocationInfo_data_typeerror_2(self):
        with self.assertRaises(TypeError):
         GetLocationInfo('True',-4.309890,4)
     # test with an incorrect longitude 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
```



colab.research.google.com/drive/1fGxc0jNBoj54wq87atbwG3QRdyFLjyEC#scrollTo=EZFTIn346WPa

Software SecurityProject.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
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 latitude data type
    def test_GetLocationInfo_data_typeerror_2(self):
        with self.assertRaises(TypeError):
          GetLocationInfo('True',-4.309890,4)
     # test with an incorrect longitude 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)
```

colab.research.google.com/drive/1fGxc0jNBoj54wq87atbwG3QRdyFLjyEC#scrollTo=EZFTIn346WPa

Software SecurityProject.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ 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(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)
```

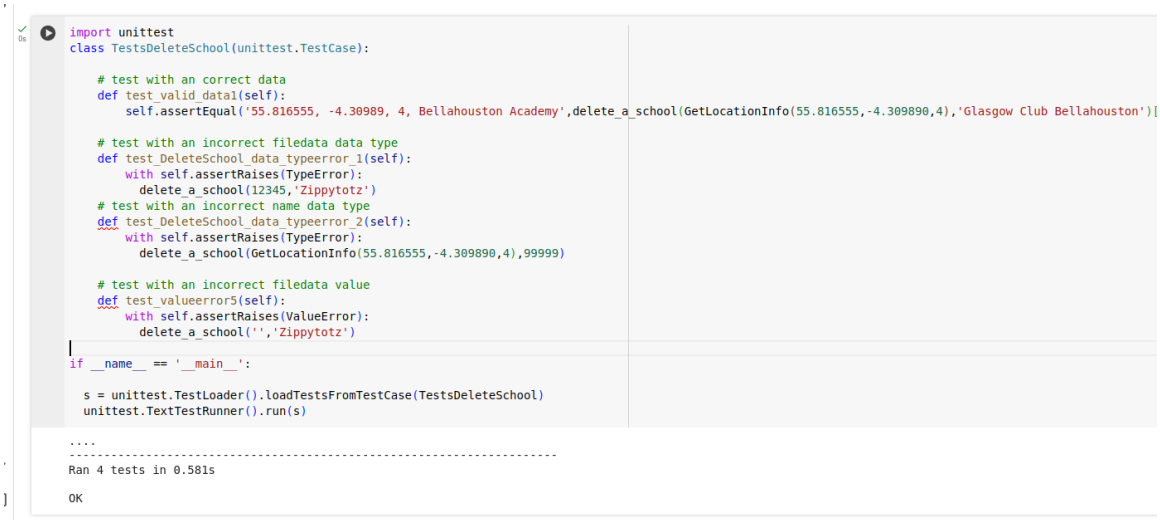
```
..............
----------------------------------------------------------------------
Ran 14 tests in 0.393s

OK
```

✓ 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 filedata 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 filedata 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
```

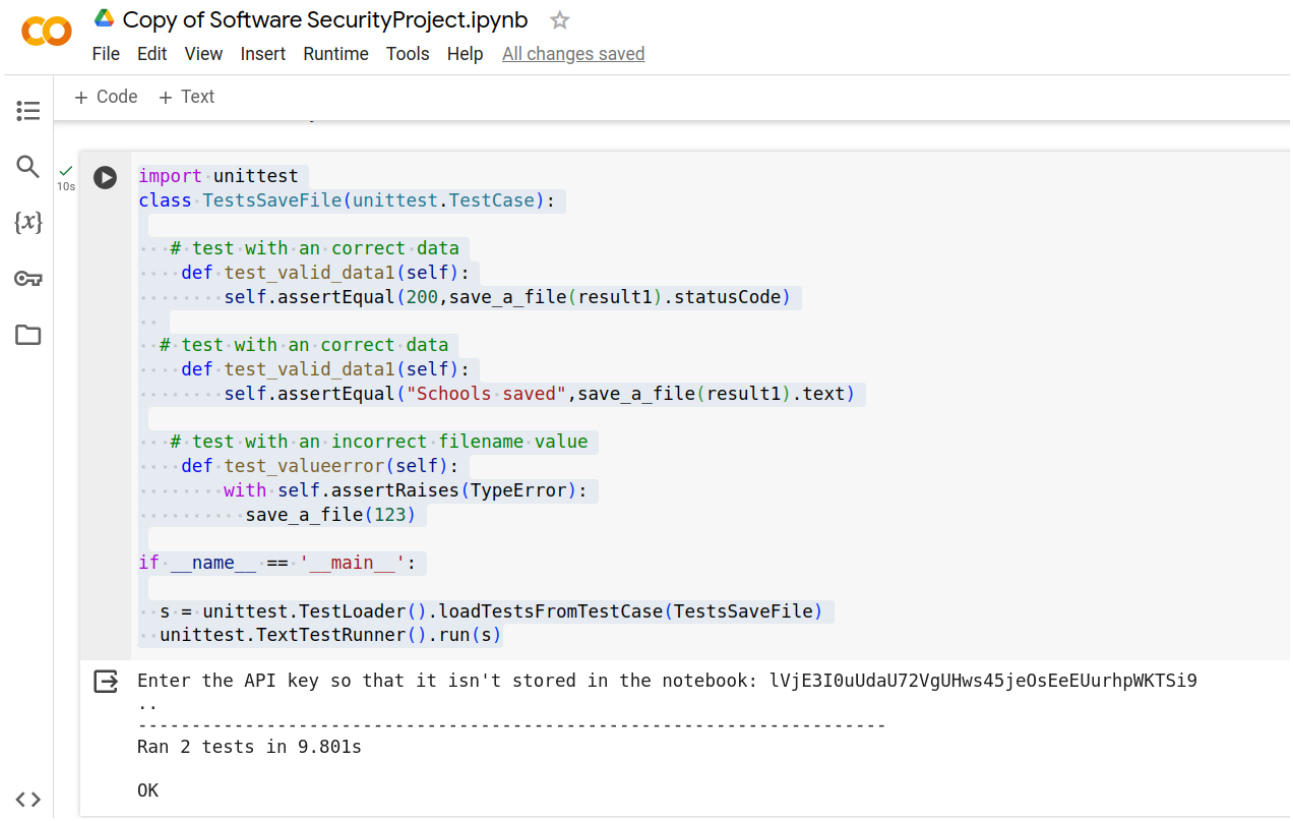


```
import unittest
class TestsDeleteSchool(unittest.TestCase):

    # 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')[

    # test with an incorrect filedata 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 filedata 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
```

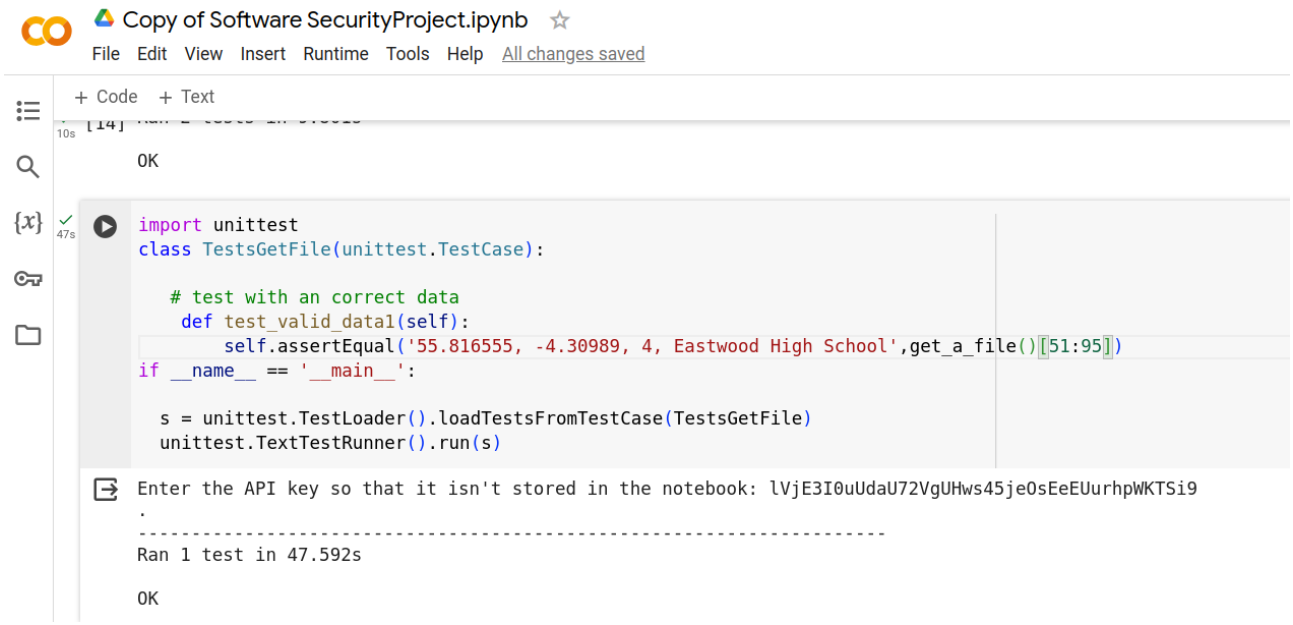
Unitests def save_a_file(filedata, filename):

```
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)
```



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)
```

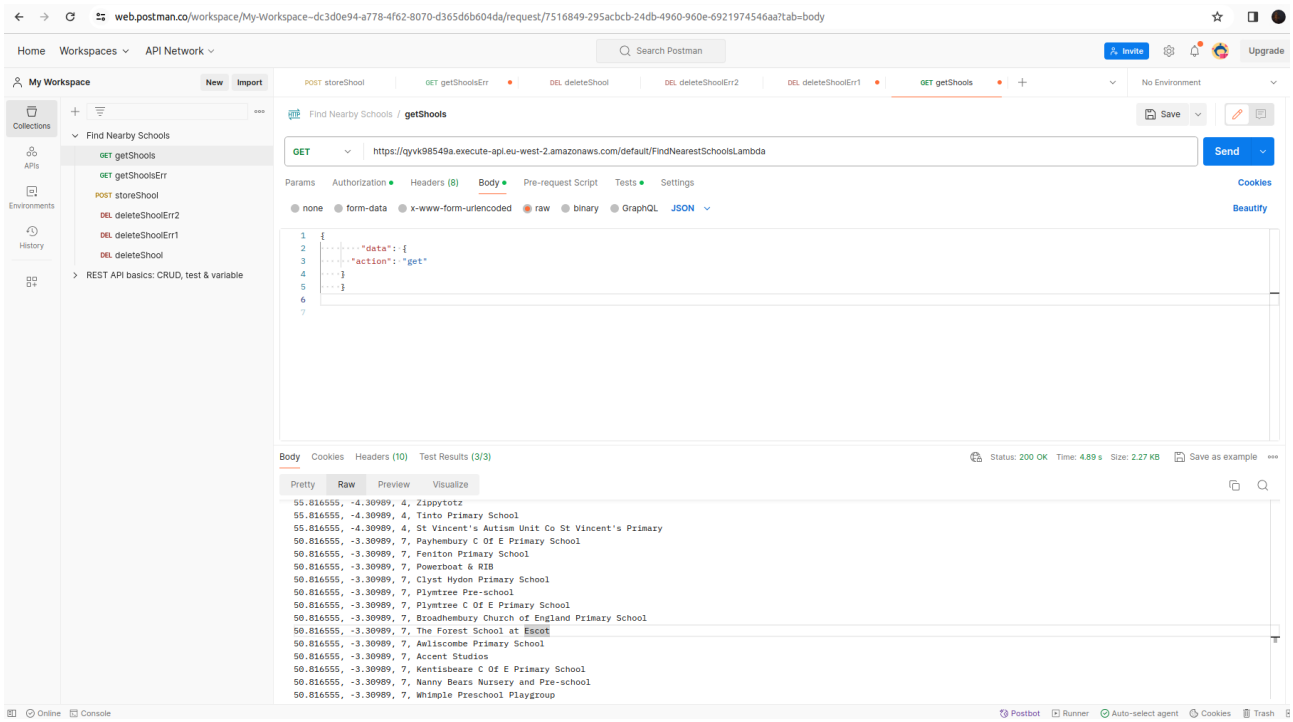


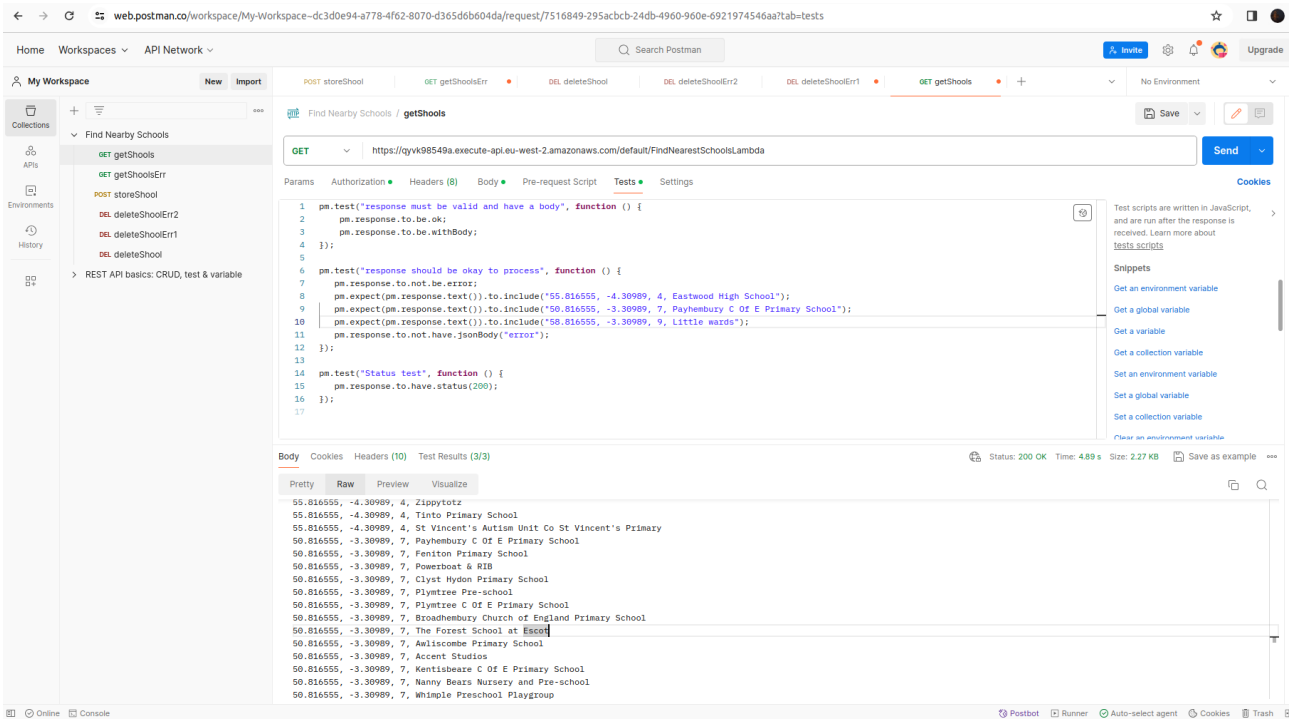
5. **Test the Lambda Function**:

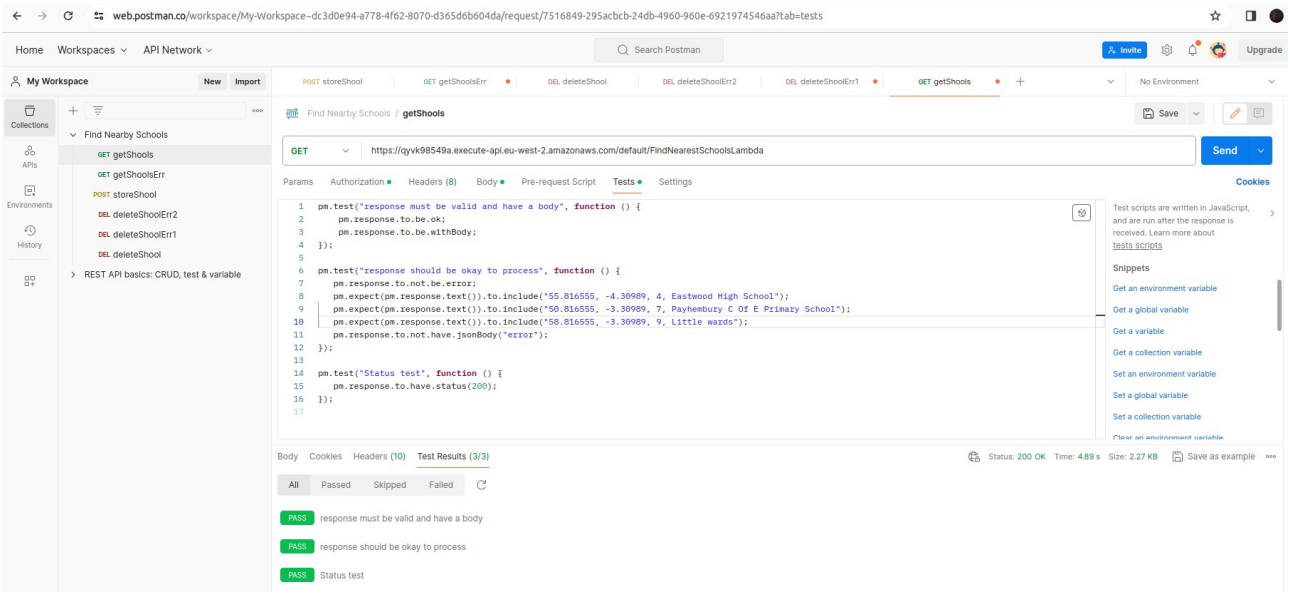
- Test the Lambda function at every stage. This means checking if it works when it's triggered by the API Gateway and if it can read and write files in S3.

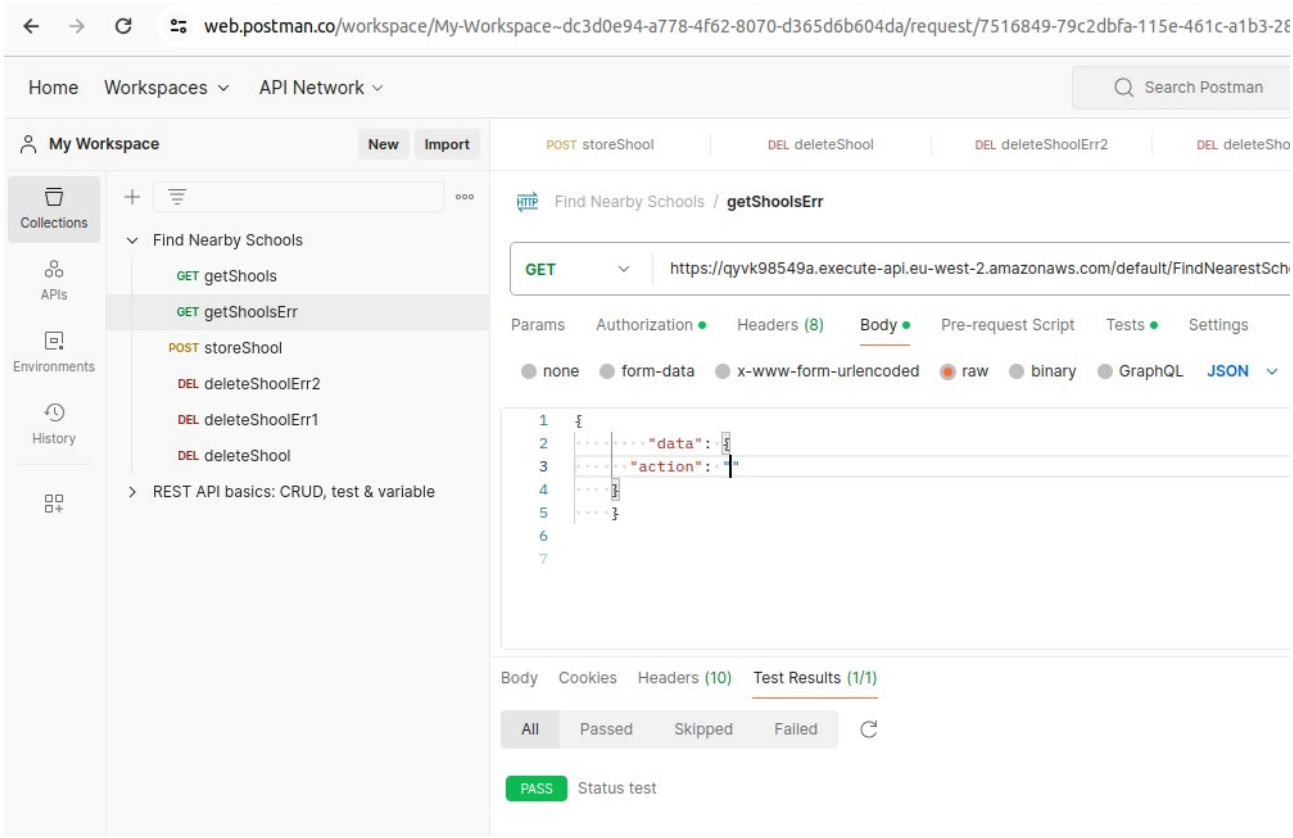
POSTMAN GET

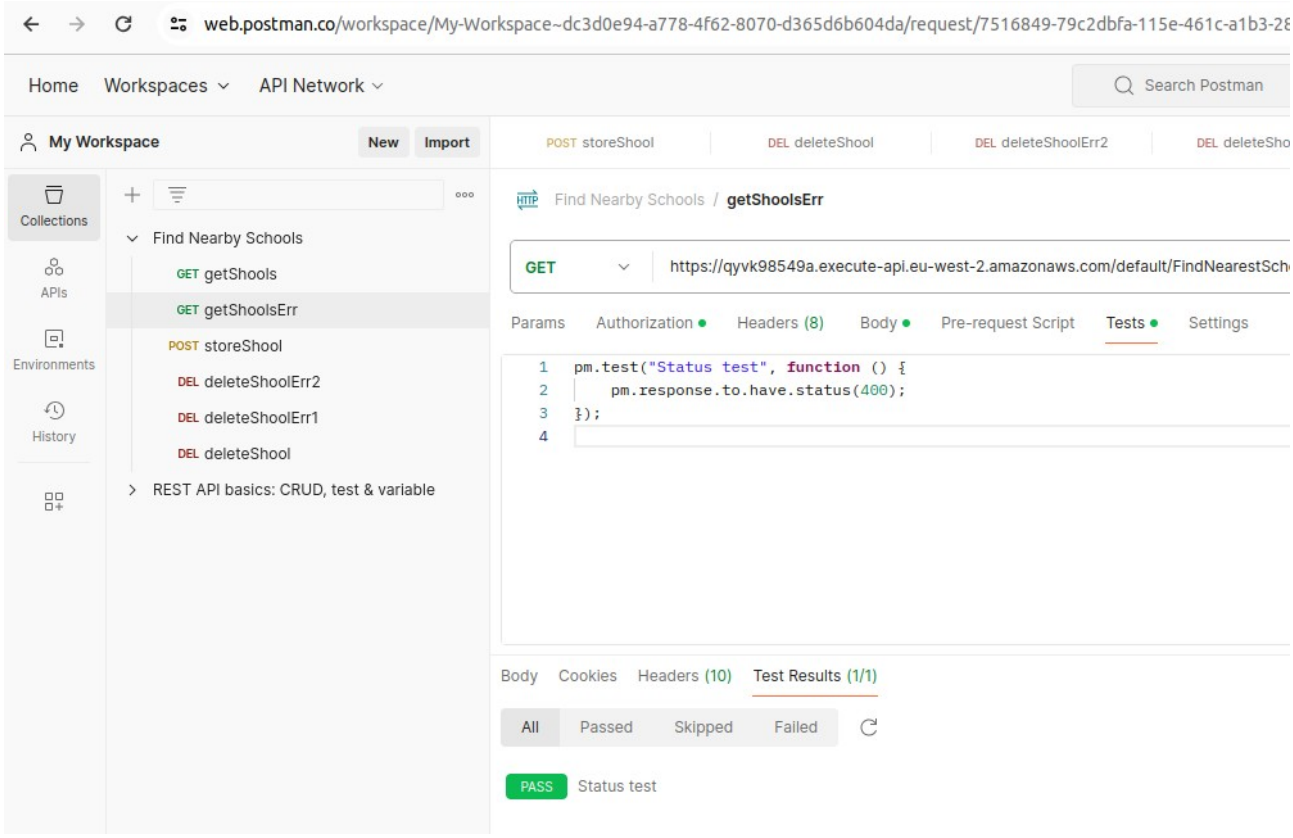
https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda




web.postman.co/workspace/My-Workspace~dc3d0e94-a778-4f62-8070-d365d6b604da/request/7516849-295acbcb-24db-4960-960e-6921974546aa?tab=tests
Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Collections
APIs
Environments
History
Find Nearby Schools
GET getShools
GET getShoolsErr
POST storeShool
DEL deleteShoolErr2
DEL deleteShoolErr1
DEL deleteShool
REST API basics: CRUD, test & variable
No Environment
Find Nearby Schools / getShools
Save
GET
https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
pm.test("response must be valid and have a body", function () {
pm.response.to.be.ok;
pm.response.to.be.withBody;
});
pm.test("response should be okay to process", function () {
pm.response.to.not.be.error;
pm.expect(pm.response.text()).to.include("55.816555, -4.30989, 4, Eastwood High School");
pm.expect(pm.response.text()).to.include("50.816555, -3.30989, 7, Payhembury C Of E Primary School");
pm.expect(pm.response.text()).to.include("58.816555, -3.30989, 9, Little wards");
pm.response.to.not.have.jsonBody("error");
});
pm.test("Status test", function () {
pm.response.to.have.status(200);
});
Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts
Snippets
Get an environment variable
Get a global variable
Get a variable
Get a collection variable
Set an environment variable
Set a global variable
Set a collection variable
Body
Cookies
Headers (10)
Test Results (3/3)
Status: 200 OK
Time: 4.89 s
Size: 2.27 KB
Save as example
Pretty
Raw
Preview
Visualize
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
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, Plymtree Pre-school
50.816555, -3.30989, 7, Plymtree C Of E Primary School
50.816555, -3.30989, 7, Broadhembury 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, Accent Studios
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, Whimple Preschool Playgroup
Online
Console
Postbot
Runner
Auto-select agent
Cookies
Trash

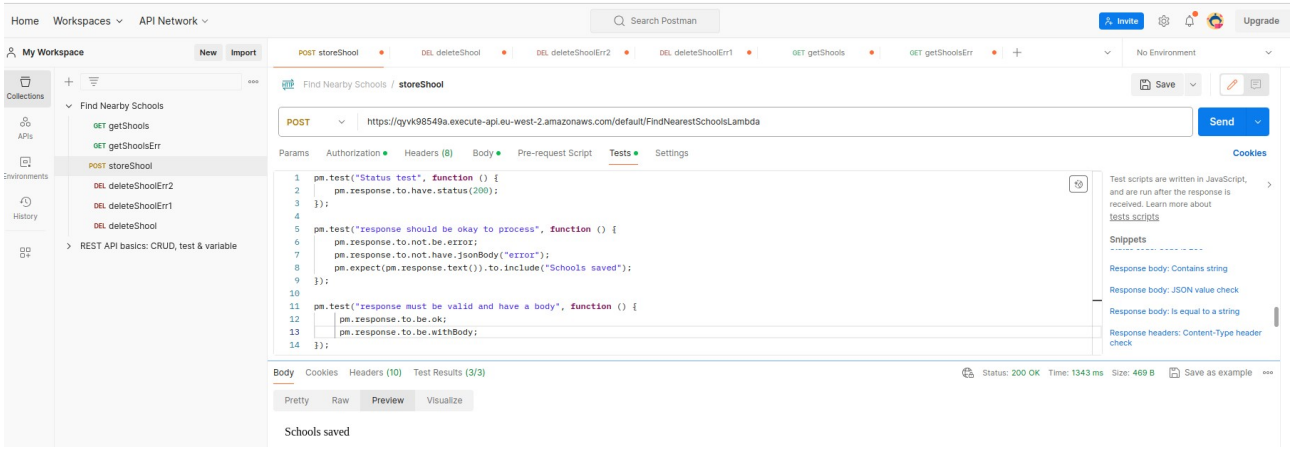

web.postman.co/workspace/My-Workspace~dc3d0e94-a778-4f62-8070-d365d6b604da/request/7516849-295acbcb-24db-4960-960e-6921974546aa?tab=tests
Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Collections
APIs
Environments
History
Find Nearby Schools
GET getShools
GET getShoolsErr
POST storeShool
DEL deleteShoolErr2
DEL deleteShoolErr1
DEL deleteShool
REST API basics: CRUD, test & variable
No Environment
Find Nearby Schools / getShools
Save
GET
https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
pm.test("response must be valid and have a body", function () {
pm.response.to.be.ok;
pm.response.to.be.withBody;
});
pm.test("response should be okay to process", function () {
pm.response.to.not.be.error;
pm.expect(pm.response.text()).to.include("55.816555, -4.30989, 4, Eastwood High School");
pm.expect(pm.response.text()).to.include("50.816555, -3.30989, 7, Payhembury C Of E Primary School");
pm.expect(pm.response.text()).to.include("58.816555, -3.30989, 9, Little wards");
pm.response.to.not.have.jsonBody("error");
});
pm.test("Status test", function () {
pm.response.to.have.status(200);
});
Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts
Snippets
Get an environment variable
Get a global variable
Get a variable
Get a collection variable
Set an environment variable
Set a global variable
Set a collection variable
Body
Cookies
Headers (10)
Test Results (3/3)
Status: 200 OK
Time: 4.89 s
Size: 2.27 KB
Save as example
All
Passed
Skipped
Failed
PASS
response must be valid and have a body
PASS
response should be okay to process
PASS
Status test

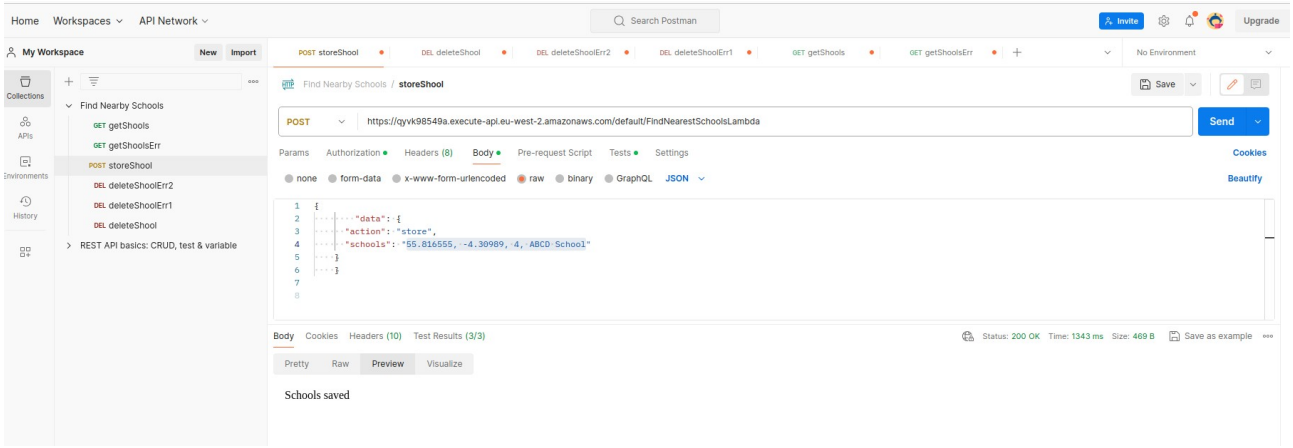

web.postman.co/workspace/My-Workspace~dc3d0e94-a778-4f62-8070-d365d6b604da/request/7516849-79c2dbfa-115e-461c-a1b3-28
Home
Workspaces
API Network
Search Postman
My Workspace
New
Import
POST storeShool
DEL deleteShool
DEL deleteShoolErr2
Collections
APIs
Environments
History
Find Nearby Schools
GET getShools
GET getShoolsErr
POST storeShool
DEL deleteShoolErr2
DEL deleteShoolErr1
DEL deleteShool
REST API basics: CRUD, test & variable
Find Nearby Schools / getShoolsErr
GET
https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSch
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"data":
"action":
}
Body
Cookies
Headers (10)
Test Results (1/1)
All
Passed
Skipped
Failed
PASS
Status test

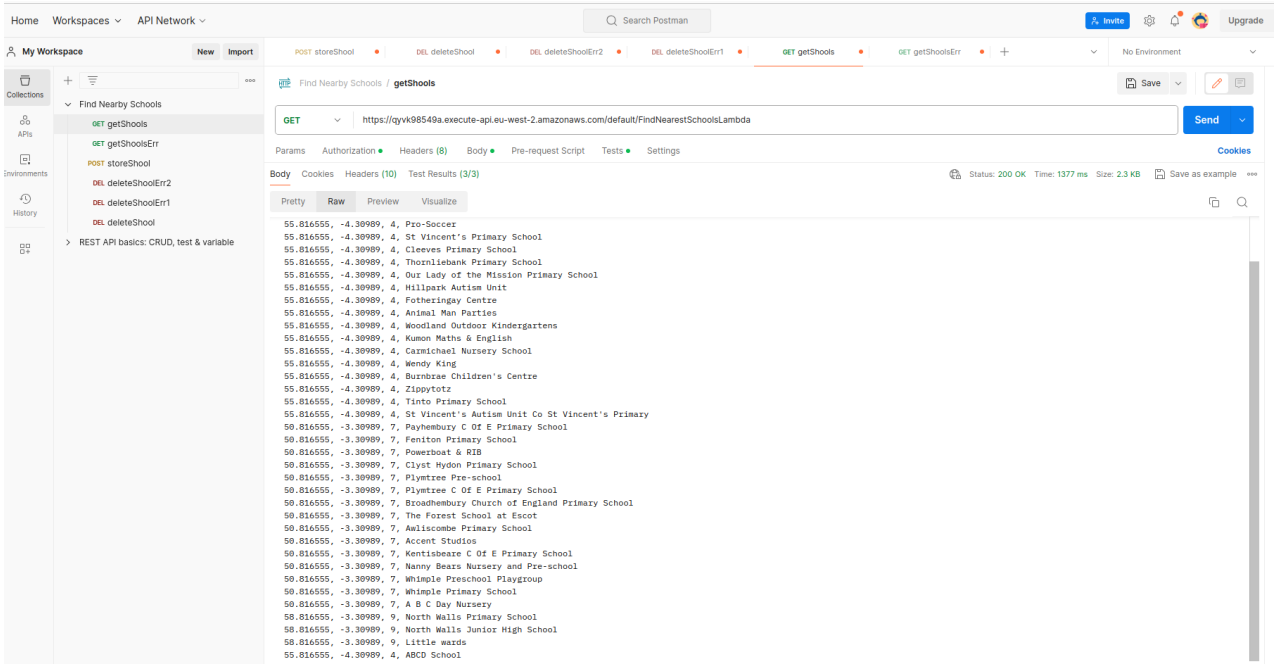

web.postman.co/workspace/My-Workspace~dc3d0e94-a778-4f62-8070-d365d6b604da/request/7516849-79c2dbfa-115e-461c-a1b3-28
Home
Workspaces
API Network
Search Postman
My Workspace
New
Import
POST storeShool
DEL deleteShool
DEL deleteShoolErr2
Collections
APIs
Environments
History
Find Nearby Schools
GET getShools
GET getShoolsErr
POST storeShool
DEL deleteShoolErr2
DEL deleteShoolErr1
DEL deleteShool
REST API basics: CRUD, test & variable
Find Nearby Schools / getShoolsErr
GET
https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSch
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
pm.test("Status test", function () {
pm.response.to.have.status(400);
});
Body
Cookies
Headers (10)
Test Results (1/1)
All
Passed
Skipped
Failed
PASS
Status test

POSTMAN POST

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

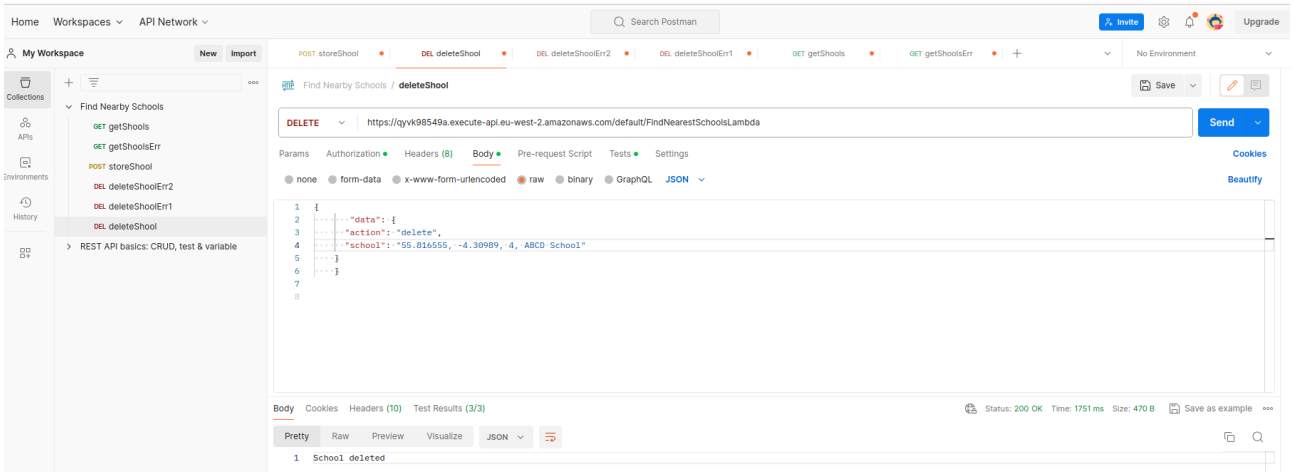


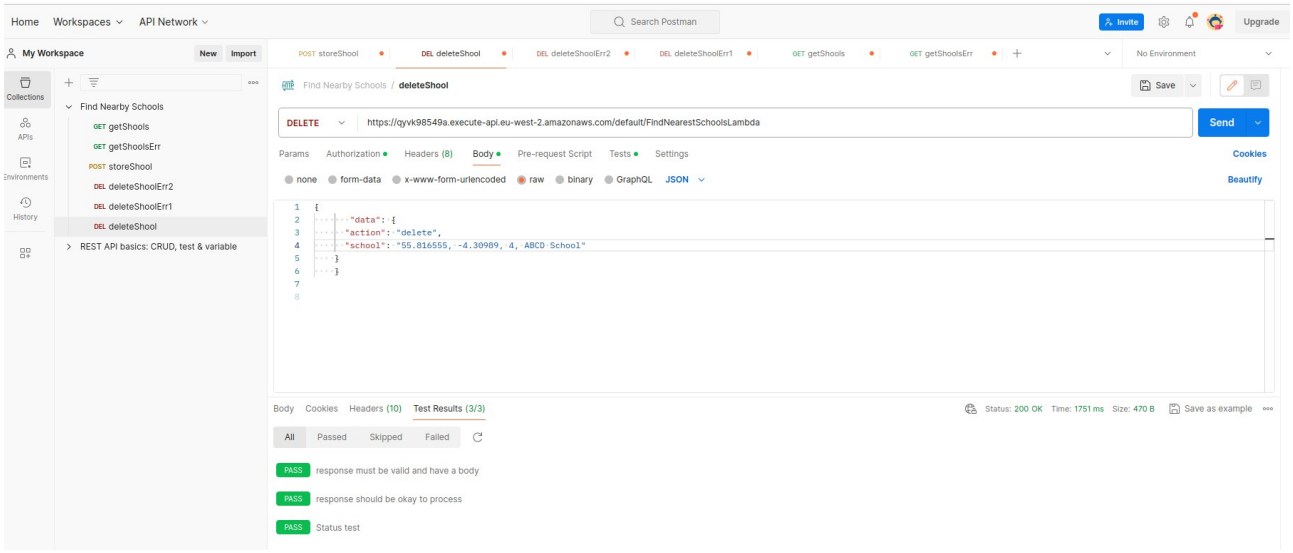


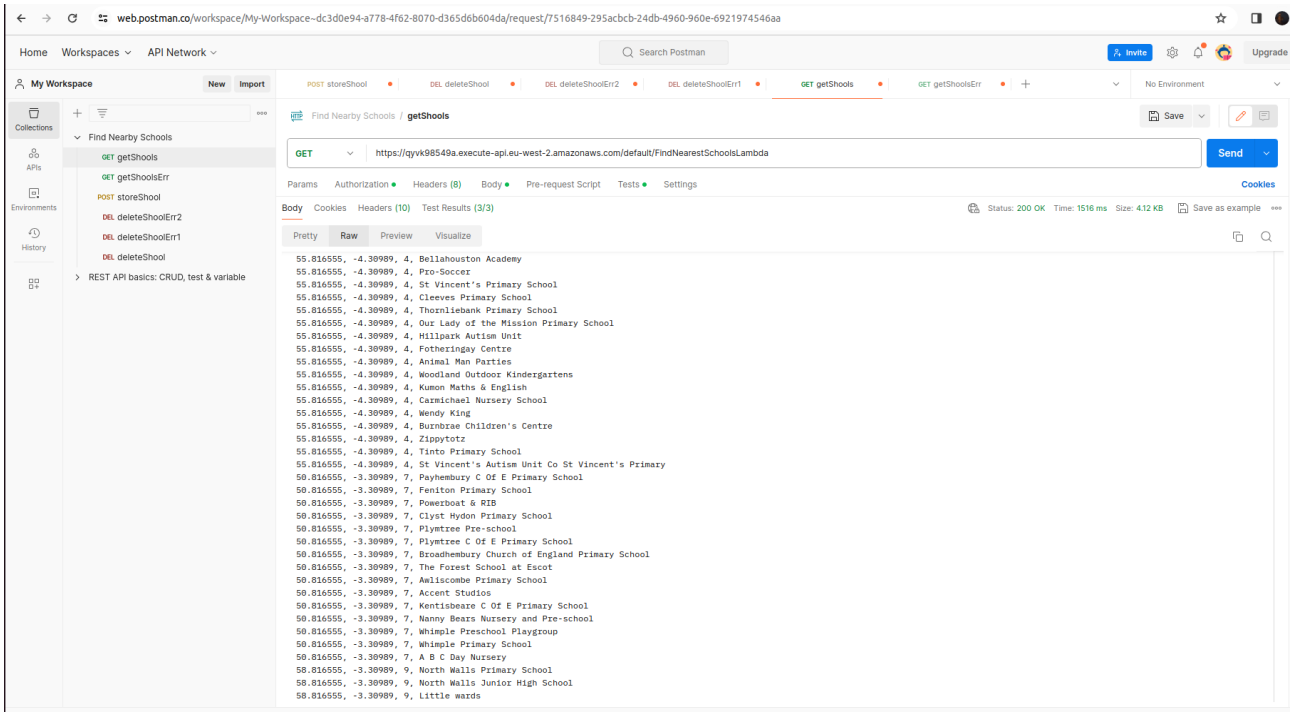


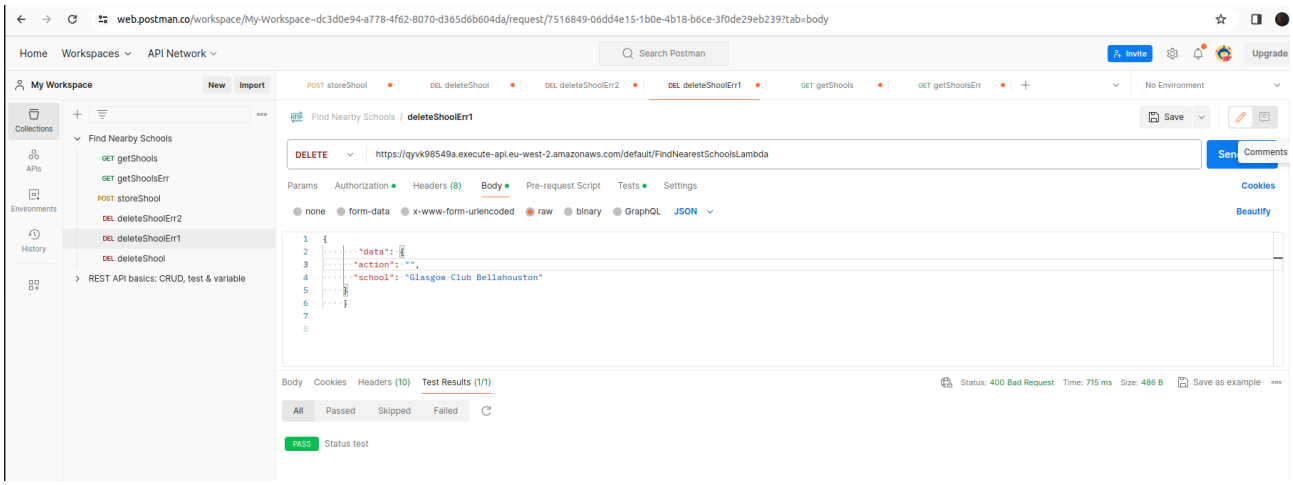
POSTMAN DELETE

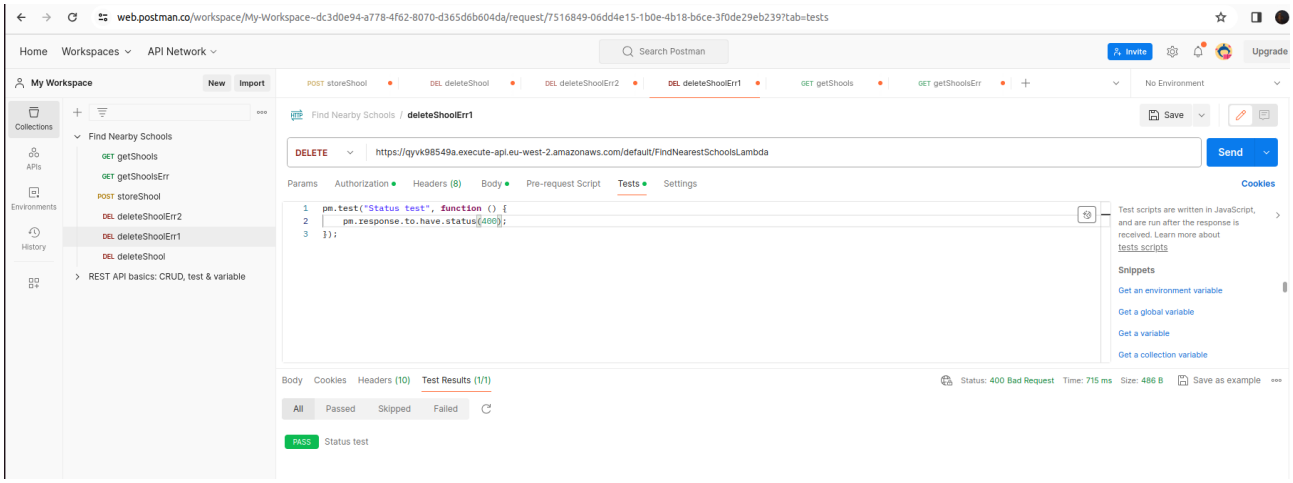
https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda

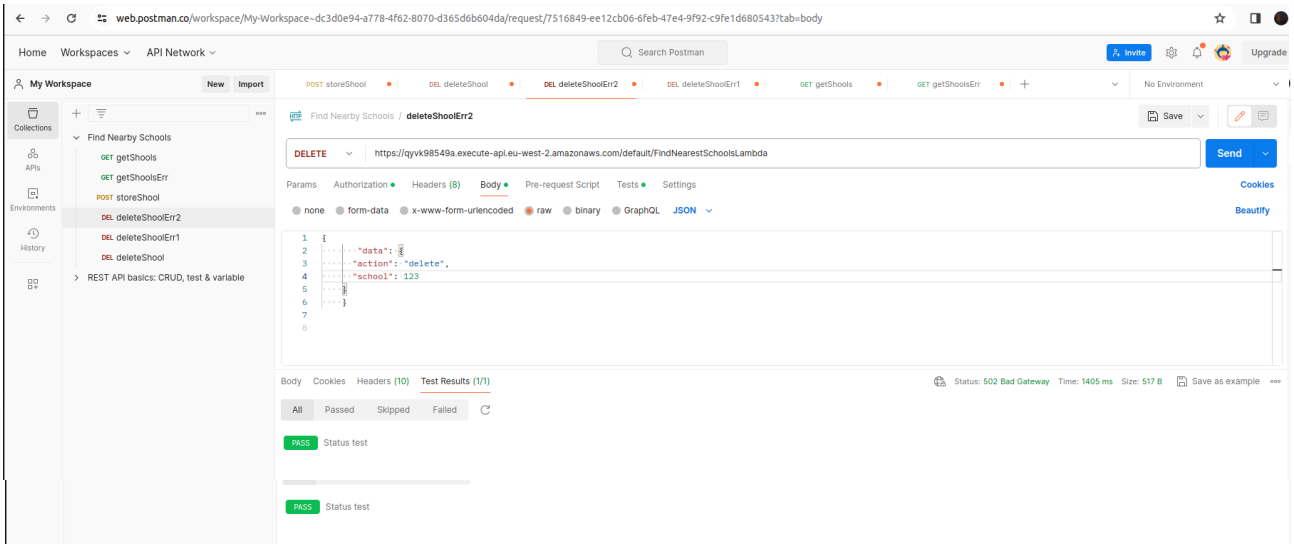


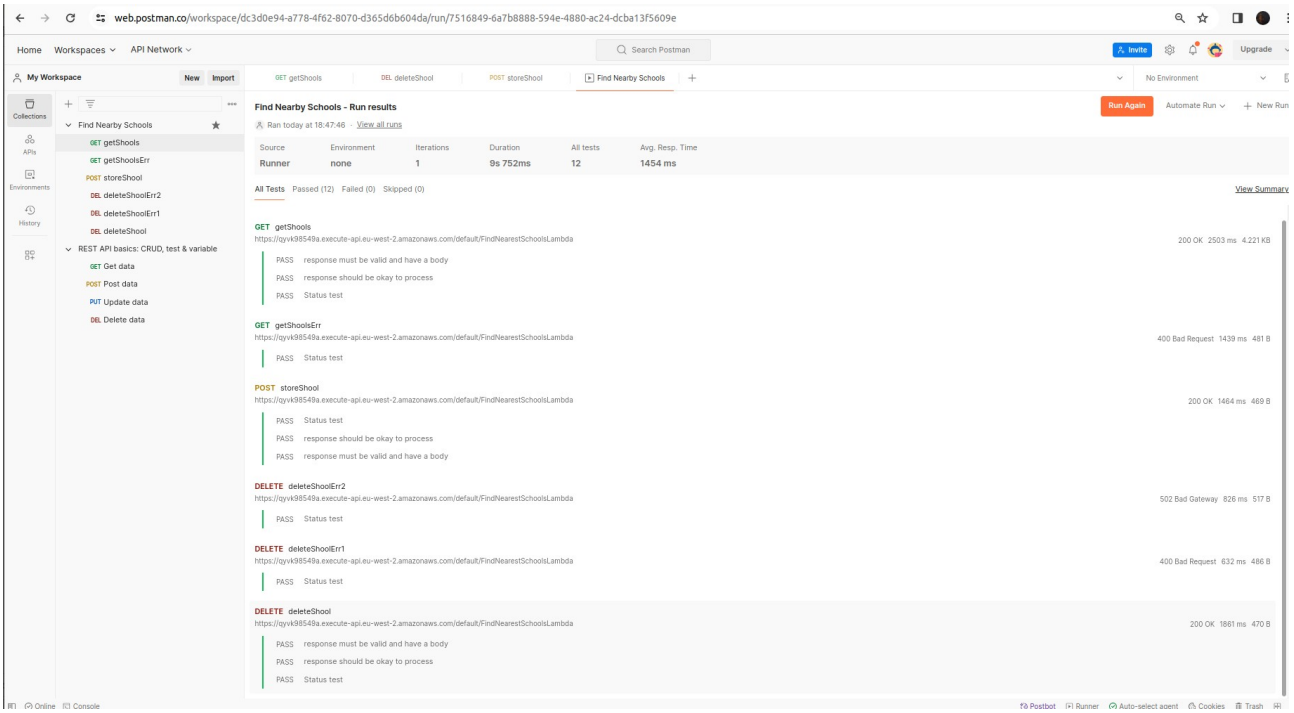


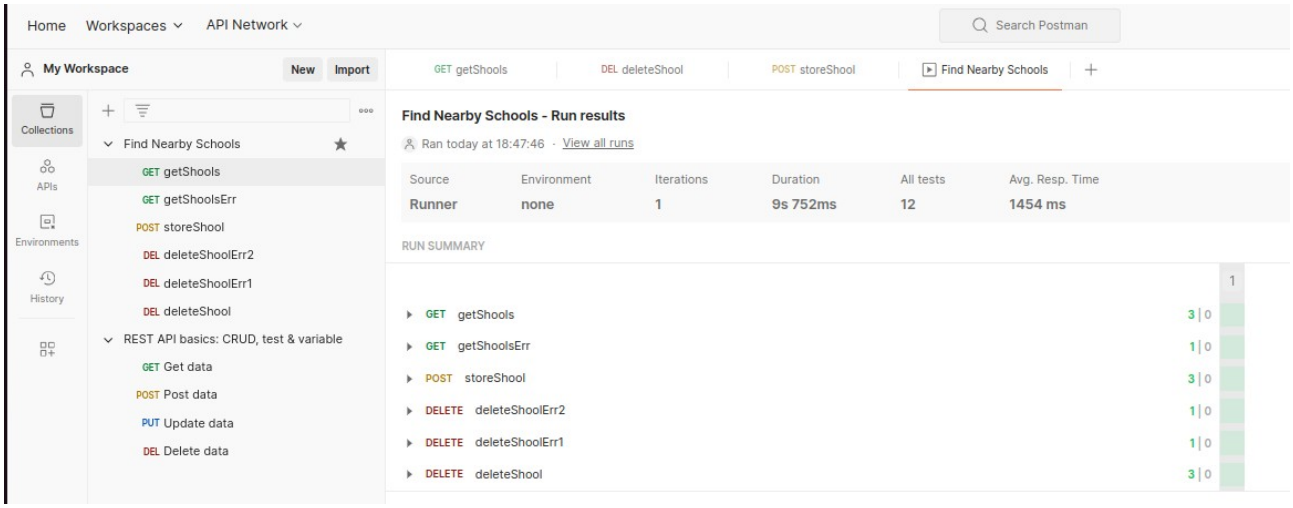


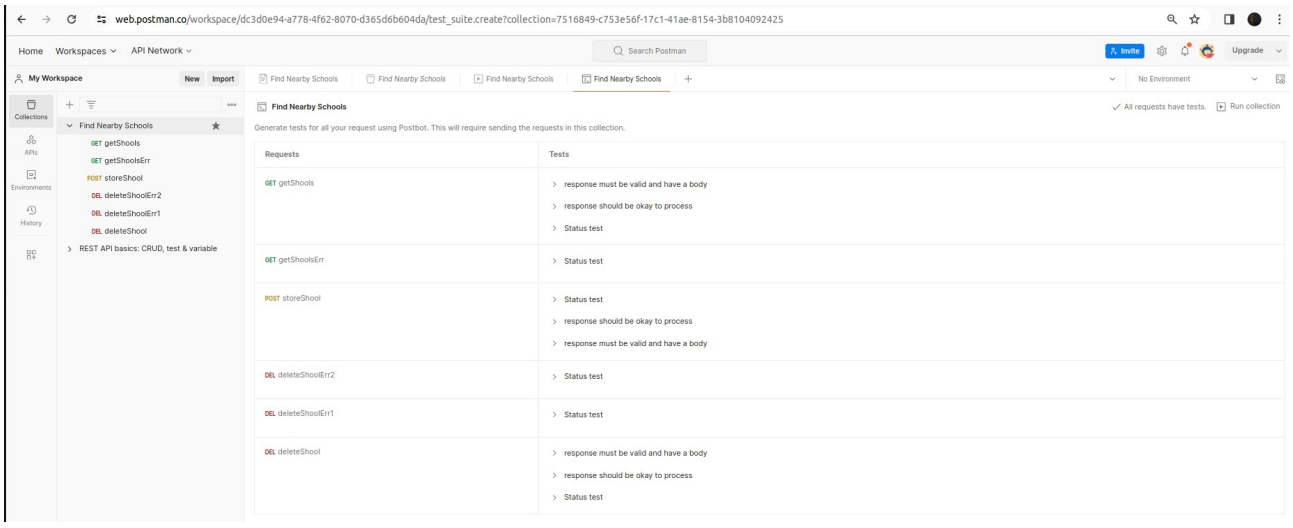

web.postman.co/workspace/My-Workspace~dc3d0e94-a778-4f62-8070-d365d6b604da/request/7516849-06dd4e15-1b0e-4b18-b6ce-3f0de29eb239?tab=body
Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST storeShool
DEL deleteShool
DEL deleteShoolErr2
DEL deleteShoolErr1
GET getShools
GET getShoolsErr
No Environment
Collections
APIs
Environments
History
Find Nearby Schools
GET getShools
GET getShoolsErr
POST storeShool
DEL deleteShoolErr2
DEL deleteShoolErr1
DEL deleteShool
REST API basics: CRUD, test & variable
Find Nearby Schools / deleteShoolErr1
Save
DELETE
https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda
Send
Comments
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"data": {
"action": "",
"school": "Glasgow Club Bellahouston"
}
}
Body
Cookies
Headers (10)
Test Results (1/1)
Status: 400 Bad Request
Time: 715 ms
Size: 486 B
Save as example
All
Passed
Skipped
Failed
PASS
Status test


web.postman.co/workspace/My-Workspace~dc3d0e94-a778-4f62-8070-d365d6b604da/request/7516849-06dd4e15-1b0e-4b18-b6ce-3f0de29eb239?tab=tests
Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST storeShool
DEL deleteShool
DEL deleteShoolErr2
DEL deleteShoolErr1
GET getShools
GET getShoolsErr
No Environment
Collections
APIs
Environments
History
Find Nearby Schools
GET getShools
GET getShoolsErr
POST storeShool
DEL deleteShoolErr2
DEL deleteShoolErr1
DEL deleteShool
REST API basics: CRUD, test & variable
Find Nearby Schools / deleteShoolErr1
Save
DELETE
https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
pm.test("Status test", function () {
pm.response.to.have.status(400);
});
Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts
Snippets
Get an environment variable
Get a global variable
Get a variable
Get a collection variable
Body
Cookies
Headers (10)
Test Results (1/1)
Status: 400 Bad Request
Time: 715 ms
Size: 486 B
Save as example
All
Passed
Skipped
Failed
PASS
Status test


web.postman.co/workspace/My-Workspace~dc3d0e94-a778-4f62-8070-d365d6b604da/request/7516849-ee12cb06-6feb-47e4-9f92-c9fe1d680543?tab=body
Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST storeShool
DEL deleteShool
DEL deleteShoolErr2
DEL deleteShoolErr1
GET getShools
GET getShoolsErr
No Environment
Collections
APIs
Environments
History
Find Nearby Schools
GET getShools
GET getShoolsErr
POST storeShool
DEL deleteShoolErr2
DEL deleteShoolErr1
DEL deleteShool
REST API basics: CRUD, test & variable
Find Nearby Schools / deleteShoolErr2
Save
DELETE
https://qyvk98549a.execute-api.eu-west-2.amazonaws.com/default/FindNearestSchoolsLambda
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"data": {
"action": "delete",
"school": 123
}
}
Body
Cookies
Headers (10)
Test Results (1/1)
Status: 502 Bad Gateway
Time: 1405 ms
Size: 517 B
Save as example
All
Passed
Skipped
Failed
PASS
Status test
PASS
Status test







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