Name = KSHITIZ GARG

Registration = 12114658

University Name = Lovely Professional University Phagwara,Punjab

Coding Interview for Graduates

Coding Challenge

Using https://api.publicapis.org/ complete the following tasks in any language/server.

1) **Create an API that lists the title, description based on the category passed as an input parameter.**

Ans:- **CODE:-**

```
const express = require('express');
const axios = require('axios');
const app = express();

app.get('/category/:name', async (req, res) => {
  try {
    const { data } = await axios.get('https://api.publicapis.org/entries');
    const categoryEntries = data.entries.filter(entry => entry.Category ===
req.params.name);
    const response = categoryEntries.map(entry => ({ title: entry.API,
description: entry.Description }));
    res.json(response);
  } catch (error) {
    res.status(500).json({ error: 'Failed to retrieve data' });
  }
});
```

DESCRIBE

To create an API that lists the title and description based on a passed category, you can follow these steps:

Create a new endpoint for your API that accepts a category as a parameter.

Use that parameter to query a database or data source that contains the relevant information (title, description, and category).

Return the results of that query in a format that is easily consumable by the client, such as JSON.

Use a framework like Flask or Django to handle the routing and logic of the API.

Test the API by sending a request to the endpoint with a specific category and verify that the correct data is returned.

2) <u>Create an API that would save a new entry with all the relevant properties which retrieves values from the endpoint GET /entries.</u>

Ans:-

CODE:

```
app.post('/save', async (req, res) => {
  try {
    const { data } = await axios.get('https://api.publicapis.org/entries');
    const newEntry = req.body;
    data.entries.push(newEntry);
    await axios.post('https://api.publicapis.org/entries', data);
    res.json({ message: 'New entry added successfully' });
  } catch (error) {
    res.status(500).json({ error: 'Failed to save new entry' });
  }
});
```

DESCRIBE

This example creates a new entry by sending a POST request to the endpoint entries with the new entry data in the request body. The new entry data is then added to the entries list. A GET request to the same endpoint will return the list of all entries.

Note that in this example there is no error handling, validation, and security features which are important in production-level APIs.

3) <u>Question: what are the key things you would consider when</u>

creating/consuming an API to ensure that it is secure and reliable?

Ans: - When creating an API, the key things to consider for security include:

1. Authentication and Authorization: ensuring that only authorized users have access to the API's resources.
2. Input validation: validating all incoming data to prevent SQL injection and other types of injection attacks.
3. Encryption: using HTTPS/TLS to encrypt all data in transit.
4. Logging and monitoring: keeping track of all API calls and monitoring for suspicious activity.

When consuming an API, the key things to consider for reliability include:

1. Error handling: Properly handling errors and exceptions
2. Caching: Caching responses from the API to improve performance and reduce the number of requests made.
3. Retry logic: Implementing retry logic to handle temporary failures or network issues.
4. Monitoring: Monitoring the API's uptime and response time to detect any issues.

When consuming an API, some key things to consider include:

1. Verifying that the API is properly authenticated and authorized for the intended user or service.
2. Validating any data received from the API to ensure that it is in the expected format and does not contain malicious content.
3. Using an up-to-date version of the API and regularly check for security updates.
4. Monitoring the API for unusual usage patterns or errors that could indicate a security issue
5. Access control and authentication: ensure that only authorized users can access the API and that their identity is verified.

   Input validation: validate all input to ensure that it is in the correct format and that it meets the requirements of the API.

   Output encoding: ensure that all output is encoded in a way that prevents cross-site scripting (XSS) attacks.

   Data encryption: ensure that sensitive data is encrypted in transit and at rest.

   Error handling: ensure that the API has robust error handling in place to prevent and handle errors gracefully.

   Testing: thoroughly test the API to ensure that it behaves as expected and that it can handle different scenarios.

   Logging and monitoring: implement logging and monitoring to keep track of how the API is being used and to detect and diagnose issues.

Theoretical Challenge

Suppose you have a CSV file with the data below.

A1: 5, A2: 7, A3: 9, B1: 3, B2: 8, B3: =4+5, C1: =5+A1, C2: =A2+B2, C3:

=C2+B3

This can be represented in an excel sheet below:

A B C

1 5 3 =5+A1

2 7 8 =A2+B2

3 9 =4+5 =C2+B3

I want a program that will take the CSV input above and produce CSV output with the results. If it is a value, then return a value. If it is a formula then calculate the formula and return the value of that formula.

1.How will you tackle the challenge above?

Ans: To tackle the challenge above, one approach would be to first read in the CSV file and parse it into a data structure such as a list of dictionaries, where each row represents a dictionary with keys as the column headers and values as the cell values. Once the data is parsed, iterate through the data structure and check if the value for each cell is a formula or a value. If it's a formula, use a library such as eval() or ast.literal_eval() to evaluate the formula and replace the formula with its calculated value. If it's a value, leave it as is. Finally, write the updated data structure to a new CSV file.

2. What type of errors you would you check for?

Ans: When checking for errors, there are several things to consider:

Syntax errors in formulas, such as using the wrong operator or forgetting to close a parenthesis

Errors in cell references, such as using a non-existent column or row in a formula

Errors in the CSV file format, such as missing headers or incorrectly formatted data

Errors in reading or writing the CSV file

3. How might a user break your code?

Ans: Some ways that a user might break the code include:

Providing an incorrectly formatted CSV file

Providing a CSV file with malicious formulas that could lead to code injection

Providing a CSV file with a large number of rows or complex formulas that could cause performance issues

Attempting to access the file system in an unauthorized way

Attempting to access a non-existent file

It is important to validate the input and handle errors properly to prevent these issues from causing unexpected behavior or security vulnerabilities.