Name = Dabbeta Ganesh Kumar

Registration = 11905491

University Name = Lovely Professional University

Coding Challenge

Ans 1:

```
from flask import Flask, jsonify

app = Flask(_name_)

 @app.route('/list/')

def list_by_category(category):

data = {'title':'title1', 'description':'description1'}

 return jsonify(data)

 if _name_ == '_main_': app.run(debug=True)
```

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.

Ans 2:

```
from flask import Flask, request

 import json app = Flask(_name_) entries = []

@app.route('/entries', methods=['GET', 'POST'])

def entry_management():

 if request.method == 'POST':

  new_entry = request.get_json()

# Append the new entry to the list of entries

entries.append(new_entry)

# Return a success message and the new entry data

 return json.dumps({'success': True, 'entry': new_entry}), 201
```

elif request.method == 'GET':

# Return the list of entries

 return json.dumps({'entries': entries}), 200

if _name_ == '_main_':

 app.run(debug=True)

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.

<u>Theoretical Challenge</u>

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

CODE:

 df=pd.Data frame(<"A":[5,7,9],"B":[3,8,9], "C":[10,15,24]})

print(df)

<u>1.How will you tackle the challenge above?</u>

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.

<u>2. What type of errors you would you check for?</u>

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

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

3. How might a user break your code?

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.Avoid unexpected security measures.