**CS301: SOFTWARE ENGINEERING**

**Semester IV**

**Class Practice Sessions - 1**

**NAME : Yadukrishna PB  
ROLL NO. : 21BCS136**

**Theme :** Create new cultural destination to celebrate the heritage of India and provide a platform for emerging Talents using Digital Technology solutions

**Aim :**

· Creating doors for a first-of-its-kind, multi-disciplinary space for the Arts in cities

· Encourage Visual art space and captivating array of public art

· Bring together communities through a dynamic programming of epic theatricals , regional theatre, music , dance , spoken word etc.

· Major attraction is to provide a platform for emerging talent and showcases the vibrance of India’s heritage

· Generate source of income for the Art communities through collaborations, aggregators and accelerators investments

**Target audiences :**

· Home to Art, Artists, the audience from India and around the world.

**Assignment scope :**

1. Identify various requirements for the above program initiative that can be developed as a digital solutions

2. Use ChatGPT platform an generate code for the above requirements

a. Generate code and run the program in Goggle Colab/Jupiter Notebook/Visual Code/PyCharm

b. Perform integrated testing. Add integration testing code in the same program.

3. Modify the same program. Write APIs to access the data from the public domain and test the program for regression testing the same program

**Deliverables :**

Working Program with test scripts embedded in the same program.

**RESULTS :**

1. Requirements :

The requirements that can be developed as a digital solutions :

* Online ticketing system: An online platform that allows users to purchase tickets for events and performances at the cultural destination. This system should include payment processing, ticket generation, and seat selection functionality.
* Virtual tours: A digital platform that allows users to explore the cultural destination and its exhibits from the comfort of their own homes. This platform should include 360-degree views, audio guides, and interactive features.
* Artist management system: A digital platform that allows artists to manage their profiles, portfolios, and bookings. This system should include tools for artists to create and edit their profiles, manage their bookings, and connect with potential collaborators.
* Collaborative workspace: A digital platform that allows artists and creatives to collaborate and work on projects remotely. This platform should include tools for sharing and collaborating on files, scheduling meetings, and tracking project progress.

**2 . CODE :**

**import unittest**

**from flask import Flask, request, jsonify**

**import json**

**import subprocess**

**app = Flask(\_\_name\_\_)**

**login\_details={}**

**art\_form\_options={}**

**def login():**

**while True:**

**username = input("Enter your username: ")**

**password = input("Enter your password: ")**

**if username == login\_details["username"] and password == login\_details["password"]:**

**print("Login successful!")**

**return True**

**else:**

**print("Invalid username or password. Please try again.")**

**# Function to display the available art forms and allow the user to choose one**

**def choose\_art\_form():**

**print("\nSelect an art form:")**

**print("1. Music")**

**print("2. Dance")**

**print("3. Spoken word")**

**print("4. Regional theatricals")**

**art\_form = int(input("Enter a number: "))**

**if art\_form == 1:**

**return view\_or\_buy("music")**

**elif art\_form == 2:**

**return view\_or\_buy("dance")**

**elif art\_form == 3:**

**return view\_or\_buy("spoken word")**

**elif art\_form == 4:**

**return view\_or\_buy("regional theatricals")**

**else:**

**print("Invalid input. Please try again.")**

**return choose\_art\_form()**

**def view\_or\_buy(art\_form\_name):**

**print("\nSelect an option for", art\_form\_name)**

**print("1. View")**

**print("2. Buy")**

**print("3. Add new view or buy option")**

**option = int(input("Enter a number: "))**

**if option == 1:**

**return view\_art\_form(art\_form\_name)**

**elif option == 2:**

**return buy\_art\_form(art\_form\_name)**

**elif option == 3:**

**return add\_option(art\_form\_name)**

**else:**

**print("Invalid input. Please try again.")**

**return view\_or\_buy(art\_form\_name)**

**def add\_option(art\_form\_name):**

**option\_name = input("Enter option name: ")**

**option\_price = input("Enter option price: ")**

**try:**

**option\_price = int(option\_price)**

**except ValueError:**

**print("Invalid price. Please try again.")**

**return add\_option(art\_form\_name)**

**art\_form\_options[art\_form\_name]['view'].append(option\_name)**

**art\_form\_options[art\_form\_name]['buy'][option\_name] = option\_price**

**print(f"Successfully added {option\_name} to {art\_form\_name} {view\_or\_buy} options.")**

**return go\_back\_option(art\_form\_name)**

**def view\_art\_form(art\_form\_name):**

**view\_options = art\_form\_options[art\_form\_name]["view"]**

**if len(view\_options) == 0:**

**print("No options available for", art\_form\_name)**

**else:**

**print("View options for", art\_form\_name, ":")**

**for option in view\_options:**

**print("-", option)**

**return go\_back\_option(art\_form\_name)**

**def buy\_art\_form(art\_form\_name):**

**buy\_options = art\_form\_options[art\_form\_name]["buy"]**

**if len(buy\_options) == 0:**

**print("No options available for", art\_form\_name)**

**else:**

**print("Buy options for", art\_form\_name, ":")**

**for option in buy\_options:**

**print("-", option)**

**option = input("Enter name of option to see buy price: ")**

**if option in buy\_options:**

**assert buy\_options[option]==art\_form\_options[art\_form\_name]["buy"][option],"inconsistent value"**

**print("Buy price for", option, "is", buy\_options[option])**

**else:**

**print("Invalid input. Please try again.")**

**return buy\_art\_form(art\_form\_name)**

**return go\_back\_option(art\_form\_name)**

**def go\_back\_option(art\_form\_name):**

**print("\nEnter 1 to go back to", art\_form\_name)**

**print("Enter 2 to exit")**

**option = int(input("Enter a number: "))**

**if option == 1:**

**return view\_or\_buy(art\_form\_name)**

**elif option == 2:**

**return True**

**else:**

**print("Invalid input. Please try again")**

**#regression testing:**

**class TestArtForms(unittest.TestCase):**

**def test\_choose\_art\_form(self):**

**self.assertEqual(choose\_art\_form(), True)**

**def test\_login(self):**

**self.assertEqual(login(), True)**

**#API:**

**@app.route('/data', methods=['GET'])**

**def get\_data():**

**# Get the data from the request**

**global login\_details,art\_form\_options**

**data = request.get\_json()**

**login\_details=data["login"]**

**art\_form\_options=data["art"]**

**inpt=input("enter 1 to perform regression testing , 2 to run program , 3 to exit")**

**if(inpt=='1'):**

**unittest.main()**

**elif inpt=='2':**

**while True:**

**choice = input("Press 1 to login, 2 to exit ")**

**if choice == "1":**

**if login():**

**# integration testing:**

**assert choose\_art\_form() == True, "Execution error"**

**elif choice == "2":**

**print("Exiting program.")**

**break**

**else:**

**print("invalid input")**

**print("program completed")**

**else:**

**print("invalid input")**

**# Return a success message**

**return jsonify({'message': 'program execution successful'})**

**app.run(debug=True)**

**'''data to pass to http://localhost:5000/data using postman:**

**{"login":{**

**"username": "my\_username",**

**"password": "my\_password"**

**},"art":{**

**"music": {**

**"view": ["rock", "jazz", "blues"],**

**"buy": {**

**"rock": 10,**

**"jazz": 15,**

**"blues": 20**

**}**

**},**

**"dance": {**

**"view": ["ballet", "tap", "salsa"],**

**"buy": {**

**"ballet": 25,**

**"tap": 20,**

**"salsa": 30**

**}**

**},**

**"spoken word": {**

**"view": ["poetry", "storytelling"],**

**"buy": {**

**"poetry": 5,**

**"storytelling": 8**

**}**

**},**

**"regional theatricals": {**

**"view": ["nautanki", "jatra"],**

**"buy": {**

**"nautanki": 12,**

**"jatra": 18**

**}**

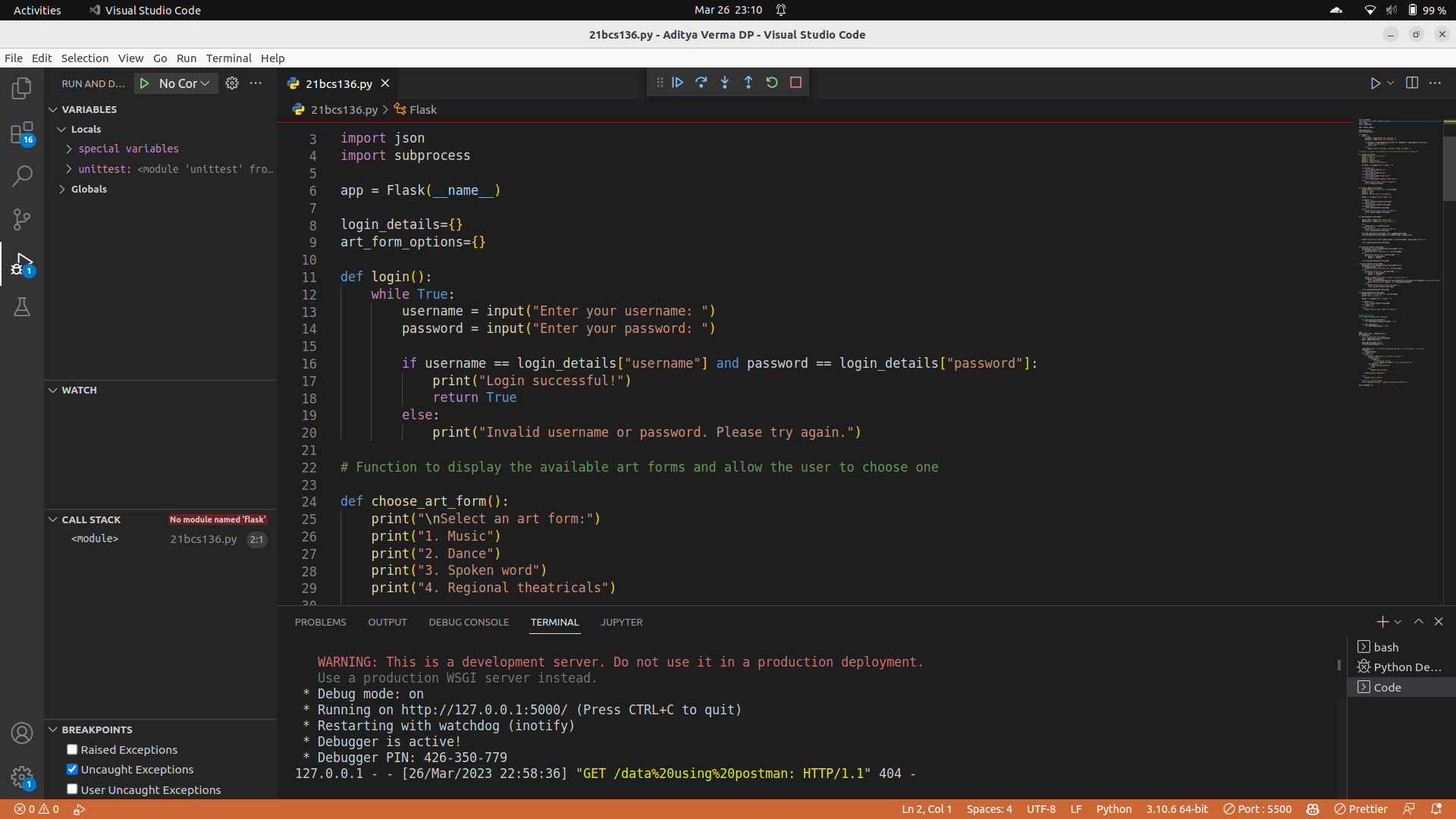
**}**

**}}**

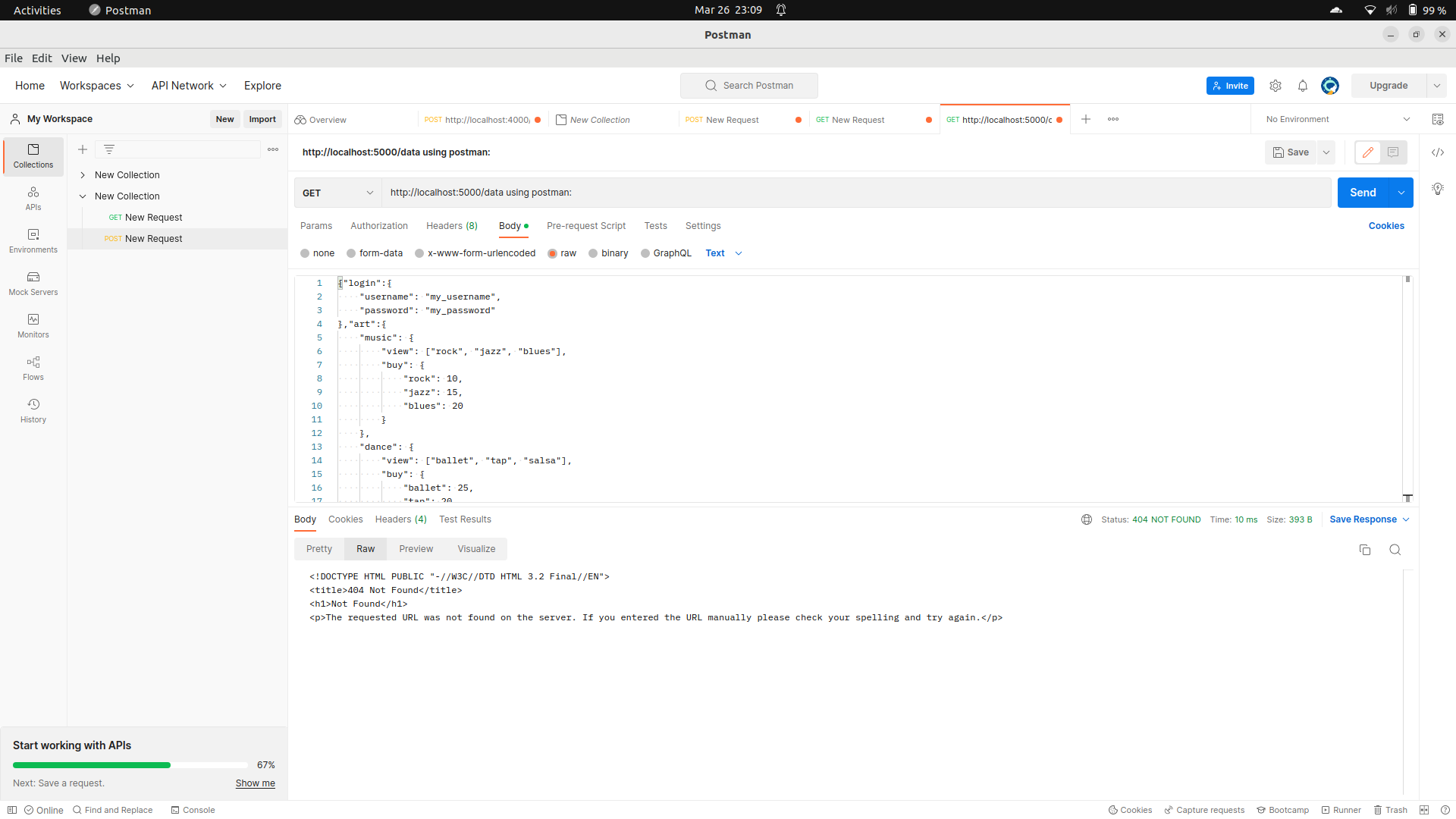
**'''**

**3.OUTPUT :**

**Code**



**API**

****

