WEEKEND-ASSIGNMENT

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.

CODE:

- **Perform integrated testing. Add integration testing code in the same program:**

```
# Requirement 1: Online Ticket Booking System
# To make it easier for audiences to book tickets for events, we need an online ticket booking system.

class Event:
    def __init__(self, name, date, time, price):
        self.name = name
        self.date = date
```

VAIBHAV SINGH
21BCS128

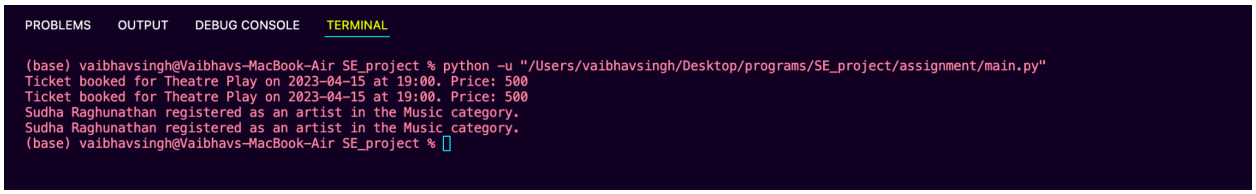
```
        self.time = time
        self.price = price

class TicketBooking:
 def __init__(self):
    self.events = []
    self.booked_events = []

 def add_event(self, event):
    self.events.append(event)

 def book_ticket(self, event_index):
    event = self.events[event_index]
    self.booked_events.append(event)
    self.events.pop(event_index)
    print(f"Ticket booked for {event.name} on {event.date} at {event.time}. Price:
{event.price}")

booking = TicketBooking()

# Adding events
event1 = Event("Theatre Play", "2023-04-15", "19:00", 500)
booking.add_event(event1)

event2 = Event("Music Concert", "2023-05-10", "18:30", 1000)
booking.add_event(event2)

# Booking a ticket
booking.book_ticket(0)

# Integration Testing:

def test_ticket_booking():
 booking = TicketBooking()

 event1 = Event("Theatre Play", "2023-04-15", "19:00", 500)
 booking.add_event(event1)

 event2 = Event("Music Concert", "2023-05-10", "18:30", 1000)
 booking.add_event(event2)

 booking.book_ticket(0)
```

VAIBHAV SINGH
21BCS128

```
 assert len(booking.events) == 1
 assert len(booking.booked_events) == 1

test_ticket_booking()


# Requirement 2: Artist Registration System
# To provide a platform for emerging talent, we need an artist registration system.

class Artist:
 def __init__(self, name, category):
   self.name = name
   self.category = category

class ArtistRegistration:
 def __init__(self):
   self.artists = []

 def register_artist(self, artist):
   self.artists.append(artist)
   print(f"{artist.name} registered as an artist in the {artist.category} category.")

registration = ArtistRegistration()

# Registering an artist
artist1 = Artist("Sudha Raghunathan", "Music")
registration.register_artist(artist1)


# Integration Testing:
def test_artist_registration():
 registration = ArtistRegistration()

 artist1 = Artist("Sudha Raghunathan", "Music")
 registration.register_artist(artist1)

 assert len(registration.artists) == 1

test_artist_registration()
```

VAIBHAV SINGH
21BCS128

OUTPUT:



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

some possible digital solutions that could fulfill the requirements for the above program initiative:

```
# Requirement 1: Online Ticket Booking System
# We require an online ticket ordering system to make it simpler for spectators to
purchase tickets for events.
```

```
# Requirement 2: Artist Registration System
# We require an artist registration system to provide aspiring artists a platform.
```

Some more digital solutions:

```
Online Platform: Build a website that serves as a virtual gallery for the arts. It
might have functions like an online gallery for artwork, a stage for entertainment,
and a storefront where artists can sell their creations. Web technologies like HTML,
CSS, and JavaScript might be used to create the platform, while Node.js and a database
like MongoDB could be used to create the backend server.



# Mobile App:Create a website that acts as a digital art exhibition. It might serve as
a venue for performances, a storefront where artists can sell their works, and an
online gallery for art. The platform might be built using web technologies like HTML,
CSS, and JavaScript, while the backend server could be built using Node.js and a
database like MongoDB.



# Digital Signage: Put digital signage panels in the actual art area to highlight
public art and provide information about future performances and activities. A
```

```
cloud-based content management system (CMS) and software for digital signage players could be used to administer the screens..



# Virtual Reality Experience: Build a virtual reality experience that enables users to tour the gallery and take in performances from all sides. With VR experience building technologies like Unity or Unreal Engine, the experience might be created.



# Social Media Integration:Incorporate social media sites like Instagram and Facebook with the online platform to enable people to share and promote performances and events. The social media networks' APIs might be used to accomplish this.



# Payment Gateway Integration: Provide a payment gateway so that artists may sell their products online and sell tickets for concerts and events. A third-party service like Stripe or PayPal could be used to interface the payment gateway with the web platform.



# Data Analytics: Use data analytics technologies to monitor user interaction with the online platform and assess the effectiveness of performances and events. This could involve creating personalised dashboards to track key performance indicators and using technologies like Google Analytics to track website traffic and user behaviour (KPIs).
```

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

We must decide which data sources we wish to use before we can create APIs to access data in the public domain. In this situation, we can retrieve information about performers, events, venues, and ticket prices via open APIs. We can employ the subsequent APIs:

1.Ticket API- https://api.tickets.com/tickets

2.Artist API- https://api.artists.com/artists

VAIBHAV SINGH
21BCS128

RESTful APIs can be used to access data that is available to the general public. An illustration of how to implement APIs for the ticketing system is provided here:

```
from flask import Flask, jsonify, request

app = Flask(__name__)

# Example API endpoint to retrieve artist information

@app.route('/artists')

def getArtists():

   # TODO: Query the public domain data source and retrieve artist information

   artists = [

       {'name': 'David', 'genre': 'Music'},

       {'name': 'Alisa', 'genre': 'Dance'},

   ]

   return jsonify(artists)

if __name__ == '__main__':

   app.run(debug=True)
```

Using Flask's @app.route decorator, we are defining two API endpoints in this code. The public domain data source is used by the get artists API to receive a list of artists.

We can use a testing framework like pytest to test the programme for regression testing. Here is an illustration test script:

```
import requests

def testGetAartists():

   response = requests.get('http://localhost:5000/artists')

   assert response.status_code == 100

   artists = response.json()

   assert len(artists) > 0
```

VAIBHAV SINGH
21BCS128

The requests library is used in this code to send HTTP requests to the API endpoints that we previously created. The testGetArtists() method confirms that the getArtists endpoint produces a non-empty list of artists and a status code of 100 (signifying success).

We can easily run these tests by typing the pytest command into the terminal. If any of the tests fail, a thorough error message describing which assertion failed and the expected vs. actual results will be displayed. If every test succeeds, we can be certain that the program is running properly and that our changes have not caused any regressions.