Software Engineering(21bcs094)

Name :- [RIDDHISH GANESH MAHAJAN IIIT Dharwad](mailto:21bcs094@iiitdwd.ac.in)

Creating a new cultural destination to honor India's history and offer a platform for up-and-coming talent using digital technology solutions is the theme of the assignment.

* The objective is to open the doors to a city's first multidisciplinary arts center.
* Encourage the creation of public art spaces and exhibits.
* a dynamic schedule of big theatricals, regional theater, music, dance, spoken word, etc. can bring communities together.
* The main draw is to give new talent a stage and display the vibrancy of India's heritage.
* Create a source of income for the art groups through investments in partnerships, aggregators, and accelerators.
* Target market: The audience from India and around the globe, as well as the home of artists.
* Mission statement: Identify potential requirements for the aforementioned programme effort.

Code:

import random

# Define the class for the cultural destination

class CulturalDestination:

def \_\_init\_\_(self, city, arts\_space, programming):

self.city = city

self.arts\_space = arts\_space

self.programming = programming

def display\_description(self):

print(f"Welcome to {self.city}! Our space for the Arts is {self.arts\_space}.")

print("We have a captivating array of public art, and a dynamic programming of epic theatricals, regional theater, music, dance, spoken word, and more.")

print("Our major attraction is providing a platform for emerging talent and showcasing the vibrance of India's heritage.")

def generate\_income(self):

print("We generate income for the Art communities through collaborations, aggregators, and accelerator investments.")

# Define the subclass for digital technology solutions

class DigitalSolutions(CulturalDestination):

def \_\_init\_\_(self, city, arts\_space, programming, technology):

super().\_\_init\_\_(city, arts\_space, programming)

self.technology = technology

def interact\_with\_technology(self):

print("Please choose from the following options:")

print("1. Take a virtual tour of the Arts space.")

print("2. Explore emerging talents in the area.")

print("3. Learn about the history and heritage of the location.")

choice = input("Enter your choice: ")

if choice == "1":

print("Loading virtual tour...")

# code to display virtual tour

elif choice == "2":

print("Exploring emerging talents...")

# code to display information on emerging talents

elif choice == "3":

print("Learning about the history and heritage...")

# code to display historical information

else:

print("Invalid choice. Please try again.")

# Create an instance of the CulturalDestination class

destination = CulturalDestination("Mumbai", "a first-of-its-kind, multi-disciplinary space for the Arts", "a dynamic programming of epic theatricals, regional theater, music, dance, spoken word, and more.")

# Display the description of the destination

destination.display\_description()

# Generate income for the Art communities

destination.generate\_income()

# Create an instance of the DigitalSolutions subclass

digital\_destination = DigitalSolutions("Mumbai", "a first-of-its-kind, multi-disciplinary space for the Arts", "a dynamic programming of epic theatricals, regional theater, music, dance, spoken word, and more.", "digital technology solutions")

# Display the description of the digital destination

digital\_destination.display\_description()

# Generate income for the Art communities using digital technology solutions

digital\_destination.generate\_income()

# Allow visitors to interact with the digital technology solutions

digital\_destination.interact\_with\_technology()

DigitalSolutions and CulturalDestination are two classes that are defined in this code. The CulturalDestination class includes characteristics for the city, the arts venue, the programming, as well as ways to present the destination's description and bring in money for the local artistic groups. The CulturalDestination class is inherited by the DigitalSolutions subclass, which also adds a way for visitors to engage with the digital technology solutions and a technology attribute.

In order to show the destination's description, fund local art communities, and enable visitors to engage with digital technology solutions, we create instances of the CulturalDestination class and the DigitalSolutions subclass, respectively, and call the methods of both classes.

* Determine different specifications for the aforementioned programme project that can be turned into digital solutions.
* Here are some possible specifications for digital products that could be created for the initiative's cultural location programme:
* a website or mobile application that offers details about the place, its opening times, its lineup, and ticketing possibilities.
* a remote exploration of the facility and its many services through a virtual tour of the arts area.
* A website that enables up-and-coming artists to display their work and make connections with prospective investors and partners.
* A social media account for the location that advertises forthcoming events, posts pictures and videos of previous performances, and promotes involvement from the local community.
* A website where consumers can buy products like books, CDs, and souvenirs that are connected to a location and its content.
* A digital archive of previous shows and exhibitions that enables viewers to revisit and share their favourite moments.
* A method for gathering and examining visitor information, such as demographics and feedback, in order to influence programming choices and marketing tactics.
* A digital marketing initiative that targets prospective tourists according to their interests, whereabouts, and demographics and tempts them to travel to the location.
* A digital map or guide that aids tourists in getting around the neighbourhood and learning about additional historical sites and tourist destinations.
* A mobile application that enables users to engage with programming components in real-time by voting for their preferred acts, asking performers questions, or sharing their

their own content.

Integrated testing should be done. Code for integration testing should be added to the programme:

Software testing known as "integrated testing" examines how various modules or components of a programme interact with one another. In the context of our program, integrated testing would entail confirming that the various digital solutions we have created function harmoniously together to deliver a consistent user experience.

As an illustration of how we might incorporate integration testing code into our programme:

import unittest

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

class TestCulturalDestination(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Chrome()

self.driver.get("https://culturaldestination.com")

def test\_homepage\_loads(self):

title = self.driver.title

self.assertEqual(title, "Cultural Destination - Celebrate the Heritage of India")

def test\_virtual\_tour(self):

self.driver.find\_element(By.ID, "virtual-tour-button").click()

WebDriverWait(self.driver, 10).until(EC.presence\_of\_element\_located((By.ID, "virtual-tour-video")))

video\_element = self.driver.find\_element(By.ID, "virtual-tour-video")

self.assertTrue(video\_element.is\_displayed())

def test\_artist\_platform(self):

self.driver.find\_element(By.ID, "artist-platform-link").click()

WebDriverWait(self.driver, 10).until(EC.presence\_of\_element\_located((By.ID, "artist-list")))

artist\_list = self.driver.find\_element(By.ID, "artist-list")

artists = artist\_list.find\_elements(By.CLASS\_NAME, "artist")

self.assertGreater(len(artists), 0)

def tearDown(self):

self.driver.quit()

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

* In this code, a test package with three test cases is defined using the Python unittest module.
* The test\_homepage\_loads function checks that the website's homepage loads properly and displays the desired headline.
* By selecting the virtual tour option and confirming that a video player loads, test\_virtual\_tour tests the functionality of the virtual tour feature.
* Test\_Artist\_Platform verifies that at least a few artists are mentioned by clicking the link for the artist platform.
* Each test scenario begins by launching the website and creating a fresh Selenium WebDriver instance. Each test case ends with a call to the tearDown function, which closes the WebDriver and frees up resources.

We only need to run the Python script in order to perform these tests. A thorough error message detailing which assertion failed and the expected vs. actual outcomes will be displayed if any of the tests fail. We can be sure that the various digital solutions we have created are interacting as intended if all tests succeed.

the same programme modified. Writing APIs to access data in the public realm and testing the programme for the same programme's regression testing

Use a Python web framework like Flask or Django to adapt the programme to include APIs for gaining access to data in the public realm. Here is an illustration using Flask:

from flask import Flask, jsonify, request

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

# Example API endpoint to retrieve artist information

@app.route('/artists')

def get\_artists():

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

artists = [

{'name': 'John Smith', 'genre': 'Folk'},

{'name': 'Jane Doe', 'genre': 'Classical'},

{'name': 'Bob Johnson', 'genre': 'Jazz'}

]

return jsonify(artists)

# Example API endpoint to retrieve event information

@app.route('/events')

def get\_events():

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

events = [

{'name': 'Summer Music Festival', 'date': '2023-07-15'},

{'name': 'Regional Dance Showcase', 'date': '2023-09-22'},

{'name': 'Winter Theater Series', 'date': '2024-01-10'}

]

return jsonify(events)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

In this code, we are defining two API endpoints using Flask's @app.route decorator. The get\_artists endpoint retrieves a list of artists from the public domain data source, while the get\_events endpoint retrieves a list of upcoming events.

To test the program for regression testing, we can use a testing framework such as pytest. Here's an example test script:

import requests

def test\_get\_artists():

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

assert response.status\_code == 200

artists = response.json()

assert len(artists) > 0

def test\_get\_events():

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

assert response.status\_code == 200

events = response.json()

assert len(events) > 0

The requests library is used in this code to make HTTP requests to the API endpoints that we previously defined. The test\_get\_artists function confirms that the get\_artists endpoint delivers a non-empty list of artists and a status code of 200 (signifying success). Similar to this, the test\_get\_events function verifies that the get\_events endpoint gives a non-empty list of events and a status code of 200.

We only need to enter the pytest command in the terminal to perform these tests. A thorough error message detailing which assertion failed and the expected vs. actual outcomes will be displayed if any of the tests fail. If every test succeeds, we can be sure that the programme is operating as intended and that our modifications did not cause any regressions.