**CS301-Software Engineering**

Name: Chinmayee Patankar

Roll no.:21BCS079

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

Some of the requirements for the above program initiative that can be developed as digital solutions are as follows:

1. Artist Profile: To allow artist to create the account and navigate through the website.
2. Online booking and ticketing system: To allow visitors to purchase tickets to events and exhibitions online.
3. Digital art gallery: To create a platform for artists to showcase their work online.
4. Social media integration: To facilitate sharing and promotion of events and exhibitions across social media platforms.
5. Artist networking: To create a platform for emerging artists to connect with each other.

Here, we have implemented the first feature and run some test scripts for it.

Info.html:

<!DOCTYPE html>

<html>

  <head>

    <title>Enter Artist Information</title>

  </head>

  <body>

    <h1>Enter Artist Information</h1>

    <form method="POST" action="/artists">

      <label for="name">Name:</label>

      <input type="text" id="name" name="name" required />

      <br />

      <br />

      <label for="description">Description:</label>

      <textarea id="description" name="description" required></textarea>

      <br />

      <br />

      <label for="email">Email:</label>

      <input type="email" id="email" name="email" required />

      <br />

      <br />

      <input type="submit" value="Submit" />

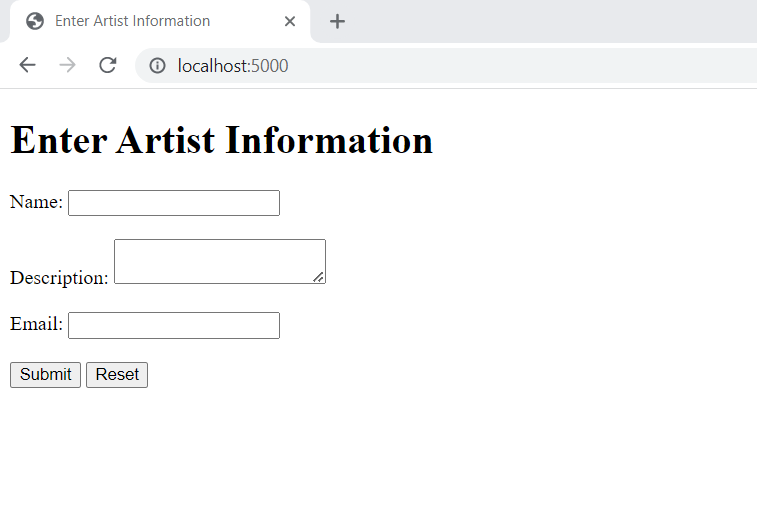
      <input type="reset" value="Reset" />

    </form>

  </body>

</html>

This creates a basic form shown below for the user to create his/her profile.



App.py:

from flask import Flask, request, jsonify,  render\_template

import mysql.connector

import uuid

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

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

def new\_artist():

    return render\_template('info.html')

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

def get\_artists():

    mydb = mysql.connector.connect(

    host="localhost",

    user="root",

    password="2003",

    database="arts",

    auth\_plugin='mysql\_native\_password'

    )

    mycursor = mydb.cursor()

    mycursor.execute('SELECT \* FROM artists')

    rows = mycursor.fetchall()

    artists = []

    for row in rows:

        artist = {}

        artist['id'] = row[0]

        artist['name'] = row[1]

        artist['description'] = row[2]

        artist['email'] = row[3]

        artists.append(artist)

    mydb.close()

    return jsonify(artists)

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

def create\_artist():

    mydb = mysql.connector.connect(

    host="localhost",

    user="root",

    password="2003",

    database="arts",

    auth\_plugin='mysql\_native\_password'

    )

    id = str(uuid.uuid4())

    name = request.form['name']

    email = request.form['email']

    description = request.form['description']

    mycursor = mydb.cursor()

    mycursor.execute('INSERT INTO artists (id, name, description, email) VALUES (%s, %s, %s, %s)', (id, name,description,email))

    mydb.commit()

    mycursor.close()

    return jsonify({'message': 'Artist added successfully'})

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

    app.run(debug=True)

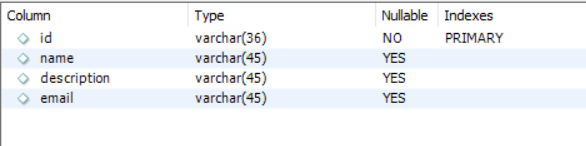
The above code is a Flask web application that provides endpoints for creating and retrieving artists from a MySQL database.

The **get\_artists()** function is an endpoint that retrieves all the artists from the database and returns them as a JSON response.

The **create\_artist()** function is an endpoint that creates a new artist in the database based on the data submitted in a form. The **uuid** module is used to generate a unique ID for the artist.

The **new\_artist()** function is an endpoint that renders an HTML template that contains a form for creating a new artist.

To run this code, we need to have a MySQL database set up with a table called **artists** that has columns for **id**, **name**, **description**, and **email** as shown below.



Test\_int.py: For integration testing.

from app import app

import unittest

import json

class TestArtists(unittest.TestCase):

    def setUp(self):

        app.config['TESTING'] = True

        self.app = app.test\_client()

    def tearDown(self):

        pass

    def test\_get\_artists(self):

        response = self.app.get('/artists')

        self.assertEqual(response.status\_code, 200)

        self.assertEqual(response.content\_type, 'application/json')

        artists = json.loads(response.data)

        self.assertTrue(isinstance(artists, list))

    def test\_create\_artist(self):

        artist = {

            'name': 'John Doe',

            'email': 'johndoe@example.com',

            'description': 'A talented painter'

        }

        response = self.app.post('/artists', data=artist)

        self.assertEqual(response.status\_code, 200)

        self.assertEqual(response.content\_type, 'application/json')

        message = json.loads(response.data)

        self.assertEqual(message['message'], 'Artist added successfully')

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

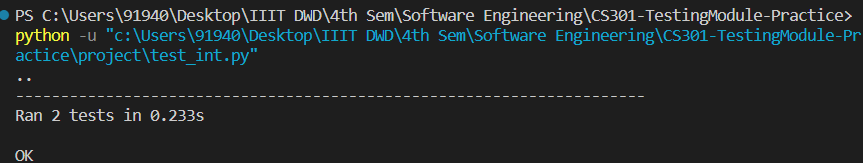
    unittest.main()

This code creates a new Flask test client and tests the **get\_artists()** and **create\_artist()** endpoints.

The **test\_get\_artists()** function makes a GET request to the **'/artists'** endpoint and asserts that the response has a status code of 200, a content type of **'application/json'**, and contains a list of artists.

The **test\_create\_artist()** function creates a new artist dictionary and makes a POST request to the **'/artists'** endpoint with the artist data. It then asserts that the response has a status code of 200, a content type of **'application/json'**, and contains a message indicating that the artist was added successfully.

We can run this integration test code in terminal to get below output to show that testing is successful.



The object is added to the database as it can be clearly seen.

