**Test Suite**: test\_app.py

import unittest

from flask\_testing import TestCase

from flask import Flask, jsonify

import json

from app import app # Import your Flask app

class TestTextToImageAPI(TestCase):

def create\_app(self):

# Return the Flask app instance

app.config['TESTING'] = True

return app

# Test if the API can generate an image with a valid prompt

def test\_generate\_image\_valid(self):

# Send a POST request to the `/generate` endpoint with valid prompt

response = self.client.post('/generate',

data=json.dumps({'prompt': 'a beautiful sunset'}),

content\_type='application/json')

# Check that the response is 200 OK

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

# Parse the response JSON

data = response.get\_json()

# Check if the response contains 'generated\_image', 'clip\_results', and 'sam\_masks'

self.assertIn('generated\_image', data)

self.assertIn('clip\_results', data)

self.assertIn('sam\_masks', data)

# Test for missing prompt (Invalid Request)

def test\_generate\_image\_missing\_prompt(self):

# Send a POST request to the `/generate` endpoint without a prompt

response = self.client.post('/generate',

data=json.dumps({}),

content\_type='application/json')

# Check that the response status is 400 Bad Request

self.assertEqual(response.status\_code, 500)

# Parse the response JSON

data = response.get\_json()

# Check if an error message is present in the response

self.assertIn('error', data)

self.assertEqual(data['error'], "Invalid input format. 'prompt' is required.")

# Test for incorrect content type

def test\_generate\_image\_incorrect\_content\_type(self):

# Send a POST request to the `/generate` endpoint with incorrect content type

response = self.client.post('/generate',

data=json.dumps({'prompt': 'a beautiful landscape'}),

content\_type='text/plain')

# Check that the response is 400 Bad Request

self.assertEqual(response.status\_code, 500)

# Test if server handles large prompt sizes

def test\_generate\_image\_large\_prompt(self):

large\_prompt = "a" \* 10000 # A prompt with 10,000 'a' characters

response = self.client.post('/generate',

data=json.dumps({'prompt': large\_prompt}),

content\_type='application/json')

# Ensure the server responds successfully

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

# Test if SAM segmentation fails gracefully

def test\_sam\_failure\_handling(self):

# Send a request with an invalid point to simulate a failure in SAM processing

response = self.client.post('/generate',

data=json.dumps({'prompt': 'a complex object'}),

content\_type='application/json')

# Check the response for errors

data = response.get\_json()

self.assertIn('error', data)

# Run the tests

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

unittest.main()

**Test Suite Explanation:**

1. **TestTextToImageAPI** Class:
   * This is the test class where each method is a test case.
   * It extends Flask-Testing's TestCase to simulate a test environment for the Flask app.
2. **Test Cases**:
   * **test\_generate\_image\_valid**: Tests that an image is generated when a valid prompt is provided. Checks for generated\_image, clip\_results, and sam\_masks in the response.
   * **test\_generate\_image\_missing\_prompt**: Tests that the API returns a 500 error when the prompt is missing.
   * **test\_generate\_image\_incorrect\_content\_type**: Tests the API when the wrong content type (text/plain instead of application/json) is provided.
   * **test\_generate\_image\_large\_prompt**: Tests how the API handles large prompts (e.g., 10,000 characters).
   * **test\_sam\_failure\_handling**: Simulates a scenario where SAM segmentation might fail and checks that the API returns an appropriate error message.