```python
# FILEPATH: /Users/defalt/Desktop/Athena/research/swarms-cloud/tests/test_cogvlm.py


import pytest

from fastapi import HTTPException

from servers.cogvlm.cogvlm import (

    create_chat_completion,

    ChatCompletionRequest,

    ChatMessageRequest,

    EventSourceResponse,

)



# Create a fixture for a valid request

@pytest.fixture

def valid_request():

    return ChatCompletionRequest(

        model="gpt-3",

        messages=[ChatMessageRequest(role="user", content="Hello")],

        temperature=0.5,

        top_p=0.5,

        max_tokens=100,

        stream=False,

    )



# Create a fixture for an invalid request
```

```python
@pytest.fixture
def invalid_request():
    return ChatCompletionRequest(
        model="gpt-3",
        messages=[ChatMessageRequest(role="assistant", content="Hello")],
        temperature=0.5,
        top_p=0.5,
        max_tokens=100,
        stream=False,
    )


# Test when the request is valid
def test_create_chat_completion_valid_request(valid_request):
    response = create_chat_completion(valid_request)
    assert response.model == valid_request.model
    assert len(response.choices) == 1
    assert response.choices[0].message.role == "assistant"
    assert response.object == "chat.completion"


# Test when the request is invalid
def test_create_chat_completion_invalid_request(invalid_request):
    with pytest.raises(HTTPException):
        create_chat_completion(invalid_request)
```

```
# Test when the request has no messages

def test_create_chat_completion_no_messages(valid_request):

    valid_request.messages = []

    with pytest.raises(HTTPException):

        create_chat_completion(valid_request)




# Test when the request has streaming enabled

def test_create_chat_completion_streaming(valid_request):

    valid_request.stream = True

    response = create_chat_completion(valid_request)

    assert isinstance(response, EventSourceResponse)




# Test when the request has a high temperature

def test_create_chat_completion_high_temperature(valid_request):

    valid_request.temperature = 1.0

    response = create_chat_completion(valid_request)

    assert response.model == valid_request.model




# Test when the request has a low temperature

def test_create_chat_completion_low_temperature(valid_request):

    valid_request.temperature = 0.0

    response = create_chat_completion(valid_request)
    assert response.model == valid_request.model



# Test when the request has a high top_p

def test_create_chat_completion_high_top_p(valid_request):

    valid_request.top_p = 1.0

    response = create_chat_completion(valid_request)

    assert response.model == valid_request.model



# Test when the request has a low top_p

def test_create_chat_completion_low_top_p(valid_request):

    valid_request.top_p = 0.0

    response = create_chat_completion(valid_request)

    assert response.model == valid_request.model



# Test when the request has a high max_tokens

def test_create_chat_completion_high_max_tokens(valid_request):

    valid_request.max_tokens = 2048

    response = create_chat_completion(valid_request)

    assert response.model == valid_request.model



# Test when the request has a low max_tokens

def test_create_chat_completion_low_max_tokens(valid_request):
    valid_request.max_tokens = 1

    response = create_chat_completion(valid_request)

    assert response.model == valid_request.model



# Test when the request has a different model

def test_create_chat_completion_different_model(valid_request):

    valid_request.model = "gpt-2"

    response = create_chat_completion(valid_request)

    assert response.model == valid_request.model
```