```
from unittest.mock import Mock, patch


import pytest
import requests


# This will be your project directory
from swarm_models.kosmos_two import Kosmos, is_overlapping


# A placeholder image URL for testing
TEST_IMAGE_URL                                                                                    =
"https://images.unsplash.com/photo-1673267569891-ca4246caafd7?auto=format&fit=crop&q=60&w
=400&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHx0b3BpYy1mZWVkfDM1fEpwZzZLaWRsLUhrfHxlbnw
wfHx8fHw%3D"



# Mock the response for the test image
@pytest.fixture
def mock_image_request():
    img_data = open(TEST_IMAGE_URL, "rb").read()
    mock_resp = Mock()
    mock_resp.raw = img_data
    with patch.object(
        requests, "get", return_value=mock_resp
    ) as _fixture:
        yield _fixture
```

```
# Test utility function
def test_is_overlapping():
    assert is_overlapping((1, 1, 3, 3), (2, 2, 4, 4)) is True
    assert is_overlapping((1, 1, 2, 2), (3, 3, 4, 4)) is False
    assert is_overlapping((0, 0, 1, 1), (1, 1, 2, 2)) is False
    assert is_overlapping((0, 0, 2, 2), (1, 1, 2, 2)) is True


# Test model initialization
def test_kosmos_init():
    kosmos = Kosmos()
    assert kosmos.model is not None
    assert kosmos.processor is not None


# Test image fetching functionality
def test_get_image(mock_image_request):
    kosmos = Kosmos()
    image = kosmos.get_image(TEST_IMAGE_URL)
    assert image is not None


# Test multimodal grounding
def test_multimodal_grounding(mock_image_request):
    kosmos = Kosmos()
    kosmos.multimodal_grounding(
        "Find the red apple in the image.", TEST_IMAGE_URL
    )
    # TODO: Validate the result if possible


# Test referring expression comprehension
def test_referring_expression_comprehension(mock_image_request):
    kosmos = Kosmos()
    kosmos.referring_expression_comprehension(
        "Show me the green bottle.", TEST_IMAGE_URL
    )
    # TODO: Validate the result if possible


# ... (continue with other functions in the same manner) ...


# Test error scenarios - Example
@pytest.mark.parametrize(
    "phrase, image_url",
    [
        (None, TEST_IMAGE_URL),
        ("Find the red apple in the image.", None),
        ("", TEST_IMAGE_URL),
        ("Find the red apple in the image.", ""),
    ],
)
def test_kosmos_error_scenarios(phrase, image_url):
    kosmos = Kosmos()
    with pytest.raises(Exception):
        kosmos.multimodal_grounding(phrase, image_url)



# ... (Add more tests for different edge cases and functionalities) ...


# Sample test image URLs
IMG_URL1                                                                                          =
"https://images.unsplash.com/photo-1696341439368-2c84b6c963bc?auto=format&fit=crop&q=60&w
=400&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHx0b3BpYy1mZWVkfDMzfEpwZzZLaWRsLUhrfHxlbnw
wfHx8fHw%3D"
IMG_URL2                                                                                          =
"https://images.unsplash.com/photo-1689934902235-055707b4f8e9?auto=format&fit=crop&q=60&w
=400&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHx0b3BpYy1mZWVkfDYzfEpwZzZLaWRsLUhrfHxlbnw
wfHx8fHw%3D"
IMG_URL3                                                                                          =
"https://images.unsplash.com/photo-1696900004042-60bcc200aca0?auto=format&fit=crop&q=60&w
=400&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHx0b3BpYy1mZWVkfDY2fEpwZzZLaWRsLUhrfHxlbnw
wfHx8fHw%3D"
IMG_URL4                                                                                          =
"https://images.unsplash.com/photo-1676156340083-fd49e4e53a21?auto=format&fit=crop&q=60&w
=400&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHx0b3BpYy1mZWVkfDc4fEpwZzZLaWRsLUhrfHxlbnw
```

```
wfHx8fHw%3D"
IMG_URL5 =
"https://images.unsplash.com/photo-1696862761045-0a65acbede8f?auto=format&fit=crop&q=80&w
=1287&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"


# Mock response for requests.get()
class MockResponse:
    @staticmethod
    def json():
        return {}

    @property
    def raw(self):
        return open("tests/sample_image.jpg", "rb")


# Test the Kosmos class
@pytest.fixture
def kosmos():
    return Kosmos()


# Mocking the requests.get() method
@pytest.fixture
def mock_request_get(monkeypatch):
    monkeypatch.setattr(
        requests, "get", lambda url, **kwargs: MockResponse()
    )


@pytest.mark.usefixtures("mock_request_get")
def test_multimodal_grounding(kosmos):
    kosmos.multimodal_grounding(
        "Find the red apple in the image.", IMG_URL1
    )


@pytest.mark.usefixtures("mock_request_get")
def test_referring_expression_comprehension(kosmos):
    kosmos.referring_expression_comprehension(
        "Show me the green bottle.", IMG_URL2
    )


@pytest.mark.usefixtures("mock_request_get")
def test_referring_expression_generation(kosmos):
    kosmos.referring_expression_generation(
        "It is on the table.", IMG_URL3
    )
```

```
@pytest.mark.usefixtures("mock_request_get")
def test_grounded_vqa(kosmos):
    kosmos.grounded_vqa("What is the color of the car?", IMG_URL4)


@pytest.mark.usefixtures("mock_request_get")
def test_grounded_image_captioning(kosmos):
    kosmos.grounded_image_captioning(IMG_URL5)


@pytest.mark.usefixtures("mock_request_get")
def test_grounded_image_captioning_detailed(kosmos):
    kosmos.grounded_image_captioning_detailed(IMG_URL1)


@pytest.mark.usefixtures("mock_request_get")
def test_multimodal_grounding_2(kosmos):
    kosmos.multimodal_grounding(
        "Find the yellow fruit in the image.", IMG_URL2
    )


@pytest.mark.usefixtures("mock_request_get")
def test_referring_expression_comprehension_2(kosmos):
    kosmos.referring_expression_comprehension(
        "Where is the water bottle?", IMG_URL3
    )



@pytest.mark.usefixtures("mock_request_get")
def test_grounded_vqa_2(kosmos):
    kosmos.grounded_vqa("How many cars are in the image?", IMG_URL4)



@pytest.mark.usefixtures("mock_request_get")
def test_grounded_image_captioning_2(kosmos):
    kosmos.grounded_image_captioning(IMG_URL2)



@pytest.mark.usefixtures("mock_request_get")
def test_grounded_image_captioning_detailed_2(kosmos):
    kosmos.grounded_image_captioning_detailed(IMG_URL3)
```