```
from unittest.mock import patch

import pytest
import torch
from PIL import Image
from transformers import FuyuImageProcessor, FuyuProcessor

from swarm_models.fuyu import Fuyu



# Basic test to ensure instantiation of class.
def test_fuyu_initialization():
    fuyu_instance = Fuyu()
    assert isinstance(fuyu_instance, Fuyu)



# Using parameterized testing for different init parameters.
@pytest.mark.parametrize(
    "pretrained_path, device_map, max_new_tokens",
    [
        ("adept/fuyu-8b", "cuda:0", 7),
        ("adept/fuyu-8b", "cpu", 10),
    ],
)
def test_fuyu_parameters(pretrained_path, device_map, max_new_tokens):
    fuyu_instance = Fuyu(pretrained_path, device_map, max_new_tokens)
```

```
    assert fuyu_instance.pretrained_path == pretrained_path
    assert fuyu_instance.device_map == device_map
    assert fuyu_instance.max_new_tokens == max_new_tokens


# Fixture for creating a Fuyu instance.
@pytest.fixture
def fuyu_instance():
    return Fuyu()


# Test using the fixture.
def test_fuyu_processor_initialization(fuyu_instance):
    assert isinstance(fuyu_instance.processor, FuyuProcessor)
    assert isinstance(
        fuyu_instance.image_processor, FuyuImageProcessor
    )


# Test exception when providing an invalid image path.
def test_invalid_image_path(fuyu_instance):
    with pytest.raises(FileNotFoundError):
        fuyu_instance("Hello", "invalid/path/to/image.png")


# Using monkeypatch to replace the Image.open method to simulate a failure.
```

```
def test_image_open_failure(fuyu_instance, monkeypatch):
    def mock_open(*args, **kwargs):
        raise Exception("Mocked failure")

    monkeypatch.setattr(Image, "open", mock_open)

    with pytest.raises(Exception, match="Mocked failure"):
        fuyu_instance(
            "Hello",

"https://plus.unsplash.com/premium_photo-1687149699194-0207c04bc6e8?auto=format&fit=crop&
q=80&w=1378&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D
%3D",
        )


# Marking a slow test.
@pytest.mark.slow
def test_fuyu_model_output(fuyu_instance):
    # This is a dummy test and may not be functional without real data.
    output = fuyu_instance(
        "Hello, my name is",

"https://plus.unsplash.com/premium_photo-1687149699194-0207c04bc6e8?auto=format&fit=crop&
q=80&w=1378&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D
%3D",
```

```
    )
    assert isinstance(output, str)


def test_tokenizer_type(fuyu_instance):
    assert "tokenizer" in dir(fuyu_instance)


def test_processor_has_image_processor_and_tokenizer(fuyu_instance):
    assert (
        fuyu_instance.processor.image_processor
        == fuyu_instance.image_processor
    )
    assert (
        fuyu_instance.processor.tokenizer == fuyu_instance.tokenizer
    )


def test_model_device_map(fuyu_instance):
    assert fuyu_instance.model.device_map == fuyu_instance.device_map


# Testing maximum tokens setting
def test_max_new_tokens_setting(fuyu_instance):
    assert fuyu_instance.max_new_tokens == 7
```

```
# Test if an exception is raised when invalid text is provided.
def test_invalid_text_input(fuyu_instance):
    with pytest.raises(Exception):
        fuyu_instance(None, "path/to/image.png")


# Test if an exception is raised when empty text is provided.
def test_empty_text_input(fuyu_instance):
    with pytest.raises(Exception):
        fuyu_instance("", "path/to/image.png")


# Test if an exception is raised when a very long text is provided.
def test_very_long_text_input(fuyu_instance):
    with pytest.raises(Exception):
        fuyu_instance("A" * 10000, "path/to/image.png")


# Check model's default device map
def test_default_device_map():
    fuyu_instance = Fuyu()
    assert fuyu_instance.device_map == "cuda:0"


# Testing if processor is correctly initialized
```

```
def test_processor_initialization(fuyu_instance):
    assert isinstance(fuyu_instance.processor, FuyuProcessor)


# Test `get_img` method with a valid image path
def test_get_img_valid_path(fuyu_instance):
    with patch("PIL.Image.open") as mock_open:
        mock_open.return_value = "Test image"
        result = fuyu_instance.get_img("valid/path/to/image.png")
    assert result == "Test image"


# Test `get_img` method with an invalid image path
def test_get_img_invalid_path(fuyu_instance):
    with patch("PIL.Image.open") as mock_open:
        mock_open.side_effect = FileNotFoundError
        with pytest.raises(FileNotFoundError):
            fuyu_instance.get_img("invalid/path/to/image.png")


# Test `run` method with valid inputs
def test_run_valid_inputs(fuyu_instance):
    with patch.object(
        fuyu_instance, "get_img"
    ) as mock_get_img, patch.object(
        fuyu_instance, "processor"
```

```
    ) as mock_processor, patch.object(
        fuyu_instance, "model"
    ) as mock_model:
        mock_get_img.return_value = "Test image"
        mock_processor.return_value = {
            "input_ids": torch.tensor([1, 2, 3])
        }
        mock_model.generate.return_value = torch.tensor([1, 2, 3])
        mock_processor.batch_decode.return_value = ["Test text"]
        result = fuyu_instance.run(
            "Hello, world!", "valid/path/to/image.png"
        )
    assert result == ["Test text"]


# Test `run` method with invalid text input
def test_run_invalid_text_input(fuyu_instance):
    with pytest.raises(Exception):
        fuyu_instance.run(None, "valid/path/to/image.png")


# Test `run` method with empty text input
def test_run_empty_text_input(fuyu_instance):
    with pytest.raises(Exception):
        fuyu_instance.run("", "valid/path/to/image.png")
```

```
# Test `run` method with very long text input
def test_run_very_long_text_input(fuyu_instance):
    with pytest.raises(Exception):
        fuyu_instance.run("A" * 10000, "valid/path/to/image.png")


# Test `run` method with invalid image path
def test_run_invalid_image_path(fuyu_instance):
    with patch.object(fuyu_instance, "get_img") as mock_get_img:
        mock_get_img.side_effect = FileNotFoundError
        with pytest.raises(FileNotFoundError):
            fuyu_instance.run(
                "Hello, world!", "invalid/path/to/image.png"
            )


# Test `__init__` method with default parameters
def test_init_default_parameters():
    fuyu_instance = Fuyu()
    assert fuyu_instance.pretrained_path == "adept/fuyu-8b"
    assert fuyu_instance.device_map == "auto"
    assert fuyu_instance.max_new_tokens == 500


# Test `__init__` method with custom parameters
```

```
def test_init_custom_parameters():
    fuyu_instance = Fuyu("custom/path", "cpu", 1000)
    assert fuyu_instance.pretrained_path == "custom/path"
    assert fuyu_instance.device_map == "cpu"
    assert fuyu_instance.max_new_tokens == 1000
```