```
from unittest.mock import Mock, patch

import pytest

from swarm_models.gemini import Gemini



# Define test fixtures
@pytest.fixture
def mock_gemini_api_key(monkeypatch):
    monkeypatch.setenv("GEMINI_API_KEY", "mocked-api-key")



@pytest.fixture
def mock_genai_model():
    return Mock()



# Test initialization of Gemini
def test_gemini_init_defaults(mock_gemini_api_key, mock_genai_model):
    model = Gemini()
    assert model.model_name == "gemini-pro"
    assert model.gemini_api_key == "mocked-api-key"
    assert model.model is mock_genai_model
```

```
def test_gemini_init_custom_params(
    mock_gemini_api_key, mock_genai_model
):
    model = Gemini(
        model_name="custom-model", gemini_api_key="custom-api-key"
    )
    assert model.model_name == "custom-model"
    assert model.gemini_api_key == "custom-api-key"
    assert model.model is mock_genai_model



# Test Gemini run method
@patch("swarms.models.gemini.Gemini.process_img")
@patch("swarms.models.gemini.genai.GenerativeModel.generate_content")
def test_gemini_run_with_img(
    mock_generate_content,
    mock_process_img,
    mock_gemini_api_key,
    mock_genai_model,
):
    model = Gemini()
    task = "A cat"
    img = "cat.png"
    response_mock = Mock(text="Generated response")
    mock_generate_content.return_value = response_mock
    mock_process_img.return_value = "Processed image"
```

```
    response = model.run(task=task, img=img)

    assert response == "Generated response"
    mock_generate_content.assert_called_with(
        content=[task, "Processed image"]
    )
    mock_process_img.assert_called_with(img=img)


@patch("swarms.models.gemini.genai.GenerativeModel.generate_content")
def test_gemini_run_without_img(
    mock_generate_content, mock_gemini_api_key, mock_genai_model
):
    model = Gemini()
    task = "A cat"
    response_mock = Mock(text="Generated response")
    mock_generate_content.return_value = response_mock

    response = model.run(task=task)

    assert response == "Generated response"
    mock_generate_content.assert_called_with(task=task)


@patch("swarms.models.gemini.genai.GenerativeModel.generate_content")
```

```
def test_gemini_run_exception(
    mock_generate_content, mock_gemini_api_key, mock_genai_model
):
    model = Gemini()
    task = "A cat"
    mock_generate_content.side_effect = Exception("Test exception")

    response = model.run(task=task)

    assert response is None



# Test Gemini process_img method
def test_gemini_process_img(mock_gemini_api_key, mock_genai_model):
    model = Gemini(gemini_api_key="custom-api-key")
    img = "cat.png"
    img_data = b"Mocked image data"

    with patch("builtins.open", create=True) as open_mock:
        open_mock.return_value.__enter__.return_value.read.return_value = (
            img_data
        )

        processed_img = model.process_img(img)

    assert processed_img == [
```

```
        {"mime_type": "image/png", "data": img_data}
    ]
    open_mock.assert_called_with(img, "rb")


# Test Gemini initialization with missing API key
def test_gemini_init_missing_api_key():
    with pytest.raises(
        ValueError, match="Please provide a Gemini API key"
    ):
        Gemini(gemini_api_key=None)


# Test Gemini initialization with missing model name
def test_gemini_init_missing_model_name():
    with pytest.raises(
        ValueError, match="Please provide a model name"
    ):
        Gemini(model_name=None)


# Test Gemini run method with empty task
def test_gemini_run_empty_task(mock_gemini_api_key, mock_genai_model):
    model = Gemini()
    task = ""
    response = model.run(task=task)
```

```
    assert response is None


# Test Gemini run method with empty image
def test_gemini_run_empty_img(mock_gemini_api_key, mock_genai_model):
    model = Gemini()
    task = "A cat"
    img = ""
    response = model.run(task=task, img=img)
    assert response is None


# Test Gemini process_img method with missing image
def test_gemini_process_img_missing_image(
    mock_gemini_api_key, mock_genai_model
):
    model = Gemini()
    img = None
    with pytest.raises(
        ValueError, match="Please provide an image to process"
    ):
        model.process_img(img=img)


# Test Gemini process_img method with missing image type
def test_gemini_process_img_missing_image_type(
```

```
    mock_gemini_api_key, mock_genai_model
):
    model = Gemini()
    img = "cat.png"
    with pytest.raises(
        ValueError, match="Please provide the image type"
    ):
        model.process_img(img=img, type=None)


# Test Gemini process_img method with missing Gemini API key
def test_gemini_process_img_missing_api_key(mock_genai_model):
    model = Gemini(gemini_api_key=None)
    img = "cat.png"
    with pytest.raises(
        ValueError, match="Please provide a Gemini API key"
    ):
        model.process_img(img=img, type="image/png")


# Test Gemini run method with mocked image processing
@patch("swarms.models.gemini.genai.GenerativeModel.generate_content")
@patch("swarms.models.gemini.Gemini.process_img")
def test_gemini_run_mock_img_processing(
    mock_process_img,
    mock_generate_content,
```

```
    mock_gemini_api_key,
    mock_genai_model,
):
    model = Gemini()
    task = "A cat"
    img = "cat.png"
    response_mock = Mock(text="Generated response")
    mock_generate_content.return_value = response_mock
    mock_process_img.return_value = "Processed image"

    response = model.run(task=task, img=img)

    assert response == "Generated response"
    mock_generate_content.assert_called_with(
        content=[task, "Processed image"]
    )
    mock_process_img.assert_called_with(img=img)


# Test Gemini run method with mocked image processing and exception
@patch("swarms.models.gemini.Gemini.process_img")
@patch("swarms.models.gemini.genai.GenerativeModel.generate_content")
def test_gemini_run_mock_img_processing_exception(
    mock_generate_content,
    mock_process_img,
    mock_gemini_api_key,
```

```
    mock_genai_model,
):
    model = Gemini()
    task = "A cat"
    img = "cat.png"
    mock_process_img.side_effect = Exception("Test exception")

    response = model.run(task=task, img=img)

    assert response is None
    mock_generate_content.assert_not_called()
    mock_process_img.assert_called_with(img=img)


# Test Gemini run method with mocked image processing and different exception
@patch("swarms.models.gemini.Gemini.process_img")
@patch("swarms.models.gemini.genai.GenerativeModel.generate_content")
def test_gemini_run_mock_img_processing_different_exception(
    mock_generate_content,
    mock_process_img,
    mock_gemini_api_key,
    mock_genai_model,
):
    model = Gemini()
    task = "A dog"
    img = "dog.png"
```

```
    mock_process_img.side_effect = ValueError("Test exception")

    with pytest.raises(ValueError):
        model.run(task=task, img=img)

    mock_generate_content.assert_not_called()
    mock_process_img.assert_called_with(img=img)


# Test Gemini run method with mocked image processing and no exception
@patch("swarms.models.gemini.Gemini.process_img")
@patch("swarms.models.gemini.genai.GenerativeModel.generate_content")
def test_gemini_run_mock_img_processing_no_exception(
    mock_generate_content,
    mock_process_img,
    mock_gemini_api_key,
    mock_genai_model,
):
    model = Gemini()
    task = "A bird"
    img = "bird.png"
    mock_generate_content.return_value = "A bird is flying"

    response = model.run(task=task, img=img)

    assert response == "A bird is flying"
```

```
    mock_generate_content.assert_called_once()
    mock_process_img.assert_called_with(img=img)


# Test Gemini chat method
@patch("swarms.models.gemini.Gemini.chat")
def test_gemini_chat(mock_chat):
    model = Gemini()
    mock_chat.return_value = "Hello, Gemini!"

    response = model.chat("Hello, Gemini!")

    assert response == "Hello, Gemini!"
    mock_chat.assert_called_once()


# Test Gemini list_models method
@patch("swarms.models.gemini.Gemini.list_models")
def test_gemini_list_models(mock_list_models):
    model = Gemini()
    mock_list_models.return_value = ["model1", "model2"]

    response = model.list_models()

    assert response == ["model1", "model2"]
    mock_list_models.assert_called_once()
```

```
# Test Gemini stream_tokens method
@patch("swarms.models.gemini.Gemini.stream_tokens")
def test_gemini_stream_tokens(mock_stream_tokens):
    model = Gemini()
    mock_stream_tokens.return_value = ["token1", "token2"]

    response = model.stream_tokens()

    assert response == ["token1", "token2"]
    mock_stream_tokens.assert_called_once()


# Test Gemini process_img_pil method
@patch("swarms.models.gemini.Gemini.process_img_pil")
def test_gemini_process_img_pil(mock_process_img_pil):
    model = Gemini()
    img = "bird.png"
    mock_process_img_pil.return_value = "processed image"

    response = model.process_img_pil(img)

    assert response == "processed image"
    mock_process_img_pil.assert_called_with(img)
```

Repeat the above tests for different scenarios or different methods in your Gemini class

until you have 15 tests in total.