```
from unittest.mock import MagicMock, patch

import pytest

from swarm_models.openai_tts import OpenAITTS


def test_openaitts_initialization():
    tts = OpenAITTS()
    assert isinstance(tts, OpenAITTS)
    assert tts.model_name == "tts-1-1106"
    assert tts.proxy_url == "https://api.openai.com/v1/audio/speech"
    assert tts.voice == "onyx"
    assert tts.chunk_size == 1024 * 1024


def test_openaitts_initialization_custom_parameters():
    tts = OpenAITTS(
        "custom_model",
        "custom_url",
        "custom_key",
        "custom_voice",
        2048,
    )
    assert tts.model_name == "custom_model"
    assert tts.proxy_url == "custom_url"
    assert tts.openai_api_key == "custom_key"

    assert tts.voice == "custom_voice"

    assert tts.chunk_size == 2048



@patch("requests.post")

def test_run(mock_post):

    mock_response = MagicMock()

    mock_response.iter_content.return_value = [b"chunk1", b"chunk2"]

    mock_post.return_value = mock_response

    tts = OpenAITTS()

    audio = tts.run("Hello world")

    assert audio == b"chunk1chunk2"

    mock_post.assert_called_once_with(

        "https://api.openai.com/v1/audio/speech",

        headers={"Authorization": f"Bearer {tts.openai_api_key}"},

        json={

            "model": "tts-1-1106",

            "input": "Hello world",

            "voice": "onyx",

        },

    )



@patch("requests.post")

def test_run_empty_task(mock_post):
    tts = OpenAITTS()
    with pytest.raises(Exception):
        tts.run("")


@patch("requests.post")
def test_run_very_long_task(mock_post):
    tts = OpenAITTS()
    with pytest.raises(Exception):
        tts.run("A" * 10000)


@patch("requests.post")
def test_run_invalid_task(mock_post):
    tts = OpenAITTS()
    with pytest.raises(Exception):
        tts.run(None)


@patch("requests.post")
def test_run_custom_model(mock_post):
    mock_response = MagicMock()
    mock_response.iter_content.return_value = [b"chunk1", b"chunk2"]
    mock_post.return_value = mock_response
    tts = OpenAITTS("custom_model")
    audio = tts.run("Hello world")
    assert audio == b"chunk1chunk2"
    mock_post.assert_called_once_with(
        "https://api.openai.com/v1/audio/speech",
        headers={"Authorization": f"Bearer {tts.openai_api_key}"},
        json={
            "model": "custom_model",
            "input": "Hello world",
            "voice": "onyx",
        },
    )


@patch("requests.post")
def test_run_custom_voice(mock_post):
    mock_response = MagicMock()
    mock_response.iter_content.return_value = [b"chunk1", b"chunk2"]
    mock_post.return_value = mock_response
    tts = OpenAITTS(voice="custom_voice")
    audio = tts.run("Hello world")
    assert audio == b"chunk1chunk2"
    mock_post.assert_called_once_with(
        "https://api.openai.com/v1/audio/speech",
        headers={"Authorization": f"Bearer {tts.openai_api_key}"},
        json={
            "model": "tts-1-1106",
            "input": "Hello world",
            "voice": "custom_voice",
        },
    )
```