```
# pytest imports
import time
from unittest.mock import Mock

import pytest

# Imports from your project
from swarms.utils import metrics_decorator



# Basic successful test
def test_metrics_decorator_success():
    @metrics_decorator
    def decorated_func():
        time.sleep(0.1)
        return [1, 2, 3, 4, 5]

    metrics = decorated_func()
    assert "Time to First Token" in metrics
    assert "Generation Latency" in metrics
    assert "Throughput:" in metrics



@pytest.mark.parametrize(
    "wait_time, return_val",
    [
```

```
        (0, []),
        (0.1, [1, 2, 3]),
        (0.5, list(range(50))),
    ],
)
def test_metrics_decorator_with_various_wait_times_and_return_vals(
    wait_time, return_val
):
    @metrics_decorator
    def decorated_func():
        time.sleep(wait_time)
        return return_val

    metrics = decorated_func()
    assert "Time to First Token" in metrics
    assert "Generation Latency" in metrics
    assert "Throughput:" in metrics


# Test to ensure that mocked time function was called and throughputs are calculated as expected
def test_metrics_decorator_with_mocked_time(mocker):
    mocked_time = Mock()
    mocker.patch("time.time", mocked_time)

    mocked_time.side_effect = [0, 5, 10, 20]
```

```
    @metrics_decorator
    def decorated_func():
        return ["tok_1", "tok_2"]

    metrics = decorated_func()
    assert (
        metrics
        == """
    Time to First Token: 5
    Generation Latency: 20
    Throughput: 0.1
    """
    )
    mocked_time.assert_any_call()


# Test to ensure that exceptions in the decorated function are propagated
def test_metrics_decorator_raises_exception():
    @metrics_decorator
    def decorated_func():
        raise ValueError("Oops!")

    with pytest.raises(ValueError, match="Oops!"):
        decorated_func()
```

```
# Test to ensure proper handling when decorated function returns non-list value
def test_metrics_decorator_with_non_list_return_val():
    @metrics_decorator
    def decorated_func():
        return "Hello, world!"

    metrics = decorated_func()
    assert "Time to First Token" in metrics
    assert "Generation Latency" in metrics
    assert "Throughput:" in metrics
```