```
from unittest.mock import Mock

import pytest

from swarms.telemetry.posthog_utils import (
    log_activity_posthog,
    posthog,
)



# Mock Posthog client
@pytest.fixture
def mock_posthog():
    return Mock()



# Mock environment variables
@pytest.fixture
def mock_env(monkeypatch):
    monkeypatch.setenv("POSTHOG_API_KEY", "test_api_key")
    monkeypatch.setenv("POSTHOG_HOST", "test_host")



# Test the log_activity_posthog decorator
def test_log_activity_posthog(mock_posthog, mock_env):
    event_name = "test_event"
```

```
    event_properties = {"test_property": "test_value"}

    # Create a test function with the decorator
    @log_activity_posthog(event_name, **event_properties)
    def test_function():
        pass

    # Call the test function
    test_function()

    # Check if the Posthog capture method was called with the expected arguments
    mock_posthog.capture.assert_called_once_with(
        "test_user_id", event_name, event_properties
    )


# Test a scenario where environment variables are not set
def test_missing_env_variables(monkeypatch):
    # Unset environment variables
    monkeypatch.delenv("POSTHOG_API_KEY", raising=False)
    monkeypatch.delenv("POSTHOG_HOST", raising=False)

    # Create a test function with the decorator
    @log_activity_posthog("test_event", test_property="test_value")
    def test_function():
        pass
```

```
    # Ensure that calling the test function does not raise errors
    test_function()


# Test the Posthog client initialization
def test_posthog_client_initialization(mock_env):
    assert posthog.api_key == "test_api_key"
    assert posthog.host == "test_host"
    assert posthog.debug is True
```