To write unit test cases for the generative AI log analyzer, we'll focus on testing the key functions like `load\_logs\_from\_folder`, `analyze\_log\_chunk`, and `analyze\_logs\_in\_folder`. The test cases will also include mocking the API responses to ensure that you can run tests without actually hitting the OpenAI API.

Here’s a set of unit test cases using Python's `unittest` framework and `unittest.mock` for mocking external dependencies:

python

import os

import unittest

from unittest.mock import patch, mock\_open

from your\_module import load\_logs\_from\_folder, analyze\_log\_chunk, analyze\_logs\_in\_folder

class TestLogAnalyzer(unittest.TestCase):

    @patch("builtins.open", new\_callable=mock\_open, read\_data="Sample log content")

    @patch("os.listdir", return\_value=["log1.log", "log2.log"])

    def test\_load\_logs\_from\_folder(self, mock\_listdir, mock\_file):

        folder\_path = "C:/Log"

        logs = load\_logs\_from\_folder(folder\_path)

        self.assertEqual(len(logs), 2)

        self.assertEqual(logs[0], "Sample log content")

        self.assertEqual(logs[1], "Sample log content")

        mock\_listdir.assert\_called\_once\_with(folder\_path)

        mock\_file.assert\_called\_with(os.path.join(folder\_path, "log1.log"), "r")

    @patch("langchain.chains.LLMChain.run", return\_value="Mock analysis result")

    def test\_analyze\_log\_chunk\_success(self, mock\_llm\_chain\_run):

#### Test analyzing a log chunk with success.

        log\_chunk = "Error occurred at line 42"

        result = analyze\_log\_chunk(log\_chunk)

        mock\_llm\_chain\_run.assert\_called\_once\_with({"log\_chunk": log\_chunk})

        self.assertEqual(result, "Mock analysis result")

    @patch("time.sleep", return\_value=None)

    @patch("langchain.chains.LLMChain.run", side\_effect=Exception("API error"))

    def test\_analyze\_log\_chunk\_failure(self, mock\_llm\_chain\_run, mock\_sleep):

    log\_chunk = "Critical error: disk failure"

        result = analyze\_log\_chunk(log\_chunk)

        self.assertEqual(mock\_llm\_chain\_run.call\_count, 5)

        self.assertEqual(result, "Log analysis failed due to repeated errors.")

    @patch("your\_module.load\_logs\_from\_folder", return\_value=["log content 1", "log content 2"])

    @patch("your\_module.analyze\_log\_chunk", side\_effect=["Result 1", "Result 2"])

    @patch("builtins.open", new\_callable=mock\_open)

    def test\_analyze\_logs\_in\_folder(self, mock\_file, mock\_analyze\_chunk, mock\_load\_logs):

        folder\_path = "C:/Log"

        analyze\_logs\_in\_folder(folder\_path)

        mock\_load\_logs.assert\_called\_once\_with(folder\_path)

        self.assertEqual(mock\_analyze\_chunk.call\_count, 2)

        mock\_file().write.assert\_called\_once\_with("Result 1\n\nResult 2")

#### Main block to run tests

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

### Explanation of Test Cases:

### 1. test\_load\_logs\_from\_folder:

* This test verifies the `load\_logs\_from\_folder` function by mocking the `os.listdir` function to simulate a folder with two `.log` files.
* It checks that the correct number of logs is loaded and their content is correct.

### 2. test\_analyze\_log\_chunk\_success:

* This test mocks the `LLMChain.run` method to simulate a successful API call to the OpenAI API.
* It verifies that the log chunk is passed correctly and the function returns the mock result.

### 3. test\_analyze\_log\_chunk\_failure:

* This test simulates repeated failures in the OpenAI API call by raising an exception.
* It verifies that the retry mechanism is triggered, and after 5 failed attempts, the function returns an error message.

### 4. test\_analyze\_logs\_in\_folder:

* This test simulates the entire process of loading logs, analyzing each log chunk, and aggregating the results into a final report.
* It checks that logs are loaded, each chunk is analyzed, and the final report is written to a file.

### Running the Tests:

We can run the tests by saving the code in a file (e.g., `test\_log\_analyzer.py`) and running the command:

python -m unittest test\_log\_analyzer.py