

# Template

```
## Repo: rag_unit_test_template
```

```
# 📁 Folder Structure
```

```
rag_unit_test_template/
```

```
├─ rag_pipeline.py
```

```
├─ retriever.py
```

```
├─ generator.py
```

```
├─ test/
```

```
│   └─ test_pipeline.py
```

```
│   └─ test_retriever.py
```

```
│   └─ test_generator.py
```

```
└─ requirements.txt
```

```
└─ README.md
```

```
# 📄 rag_pipeline.py
```

```
class RAGPipeline:
```

```
    def __init__(self, retriever, generator):
```

```
        self.retriever = retriever
```

```
        self.generator = generator
```

```
    def run(self, query):
```

```
        docs = self.retriever.retrieve(query)
```

```
        return self.generator.generate(query, docs)
```

```
# 📄 retriever.py
```

```
class SimpleRetriever:
```

```
    def __init__(self, corpus):
```

```
        self.corpus = corpus
```

```
    def retrieve(self, query):
```

```
return [doc for doc in self.corpus if query.lower().split()[0] in doc.lower()]
```

```
# test/generator.py
```

```
class DummyGenerator:
```

```
    def generate(self, query, documents):
```

```
        return f"Answering '{query}' using: {documents[0]}"
```

```
# test/test_pipeline.py
```

```
import pytest
```

```
from unittest.mock import MagicMock
```

```
from rag_pipeline import RAGPipeline
```

```
def test_rag_pipeline_returns_expected_output():
```

```
    mock_retriever = MagicMock()
```

```
    mock_retriever.retrieve.return_value = ["Mock doc"]
```

```
    mock_generator = MagicMock()
```

```
    mock_generator.generate.return_value = "Mock answer"
```

```
    pipeline = RAGPipeline(mock_retriever, mock_generator)
```

```
    result = pipeline.run("What is mock?")
```

```
    assert result == "Mock answer"
```

```
    mock_retriever.retrieve.assert_called_once_with("What is mock?")
```

```
    mock_generator.generate.assert_called_once_with("What is mock?", ["Mock doc"])
```

```
# test/test_retriever.py
```

```
from retriever import SimpleRetriever
```

```
def test_simple_retriever_returns_match():
```

```
    retriever = SimpleRetriever(["Paris is the capital of France."])
```

```
    docs = retriever.retrieve("What is the capital of France?")
```

```
    assert any("Paris" in doc for doc in docs)
```

```
# test/test_generator.py
```

```
from generator import DummyGenerator
```

```
def test_dummy_generator_formats_output():
```

```
    gen = DummyGenerator()
```

```
    result = gen.generate("What is AI?", ["AI is artificial intelligence."])
```

```
    assert "What is AI?" in result
```

```
    assert "AI is artificial intelligence." in result
```

```
# requirements.txt
```

```
pytest
```

```
# README.md
```

```
# RAG Unit Test Template
```

This repo demonstrates how to build a unit-tested RAG (Retrieval-Augmented Generation) system.

```
## Components
```

- SimpleRetriever: Retrieves documents based on keyword match
- DummyGenerator: Generates answer using context
- RAGPipeline: Combines retriever + generator

```
## Unit Tests
```

Run all tests with:

```
```bash
```

```
pytest test/ -v
```

```
```
```

You can expand this repo with real vector stores (FAISS/Chroma) or LLMs (Flan-T5) later.

---

Phiên bản #1

Được tạo 7 tháng 5 2025 16:04:10 bởi Đỗ Ngọc Tú

Được cập nhật 7 tháng 5 2025 16:04:37 bởi Đỗ Ngọc Tú