Full-Stack RAG with Local LLM

Laboratory Manual

CSI403 - Full Stack Development
Faculty of Information Technology, Sripatum University

Academic Year 2568

Contents

1 Lab 1: Git + Python Fundamentals (3.75%)

1.1 Objectives

- Fork and clone Generic-RAG repository
- Practice Git workflow
- Review Python fundamentals

1.2 Tasks

1.2.1 Task 1: Fork Repository

1. Go to https://github.com/amornpan/Generic-RAG
2. Click "Fork" button
3. You now have your own copy

1.2.2 Task 2: Clone to Local

```
git clone https://github.com/YOUR_USERNAME/Generic-RAG.git
cd Generic-RAG
```

1.2.3 Task 3: Setup Environment

```
conda create -n rag_env python=3.10 -y
conda activate rag_env
pip install -r requirements.txt
```

1.2.4 Task 4: Create Feature Branch

```
git checkout -b feature/lab01-python
```

1.2.5 Task 5: Write Python OOP Example

Create `lab01_example.py`:

```
class Document:
    def __init__(self, title: str, content: str):
        self.title = title
        self.content = content

    def get_summary(self) -> str:
        return self.content[:100] + "..."

# Test
doc = Document("Test", "This is a test document...")
print(doc.get_summary())
```

1.2.6 Task 6: Commit and Push

```
git add .
git commit -m "Lab 1: Add Python OOP example"
git push origin feature/lab01-python
```

1.2.7 Task 7: Create Pull Request

1. Go to GitHub
2. Click "Compare & pull request"
3. Create PR

1.3 Deliverables

Forked repository

Feature branch created

Python file committed

Pull Request created

Screenshot of PR

Deadline: Sunday 23:59

2 Lab 2: Docker + OpenSearch (3.75%)

2.1 Objectives

- Install Docker Desktop
- Run OpenSearch container
- Configure Hybrid Search Pipeline

2.2 Tasks

2.2.1 Task 1: Install Docker Desktop

1. Download from https://docker.com
2. Install and start
3. Verify: `docker -version`

2.2.2 Task 2: Run OpenSearch

```
docker run -d --name opensearch-node \
  -p 9200:9200 -p 9600:9600 \
  -e "discovery.type=single-node" \
  -e "DISABLE_SECURITY_PLUGIN=true" \
  opensearchproject/opensearch:2.11.1
```

2.2.3 Task 3: Verify

```
curl http://localhost:9200
```

2.2.4 Task 4: Setup Hybrid Search Pipeline

```
curl -X PUT "localhost:9200/_search/pipeline/hybrid-search-pipeline" \
  -H "Content-Type: application/json" \
  -d '{
    "phase_results_processors": [{
      "normalization-processor": {
        "normalization": {"technique": "min_max"},
        "combination": {
          "technique": "arithmetic_mean",
          "parameters": {"weights": [0.3, 0.7]}
        }
      }
    }]
  }'
```

2.2.5 Task 5: Check Health

```
curl http://localhost:9200/_cluster/health?pretty
```

2.3 Deliverables

Docker installed

OpenSearch running

Pipeline created

Screenshots

Deadline: Sunday 23:59

3 Lab 3: FastAPI (3.75%)

3.1 Objectives

- Study api.py from Generic-RAG
- Run FastAPI server
- Test API endpoints

3.2 Tasks

3.2.1 Task 1: Study api.py

Review the structure:

- FastAPI app
- Pydantic models
- Endpoints: /health, /search, /query

3.2.2 Task 2: Run Server

```
cd Generic-RAG
conda activate rag_env
python api.py
```

3.2.3 Task 3: Test Endpoints

```
# Health check
curl http://localhost:9000/health

# Search
curl -X POST http://localhost:9000/search \
  -H "Content-Type: application/json" \
  -d '{"query": "test", "top_k": 5}'
```

3.2.4 Task 4: Access Swagger Docs

Open: http://localhost:9000/docs

3.2.5 Task 5: Add New Endpoint

Add `/documents` endpoint to list all documents.

3.3 Repository

https://github.com/amornpan/Generic-RAG

3.4 Deliverables

Server running

Endpoints tested

New endpoint added

Screenshots

Deadline: Sunday 23:59

4 Lab 4: OpenSearch Integration (3.75%)

4.1 Objectives

- Connect to OpenSearch from Python
- Implement hybrid search
- Index and search documents

4.2 Tasks

4.2.1 Task 1: Connect to OpenSearch

```
from opensearchpy import OpenSearch

client = OpenSearch(
    hosts=[{"host": "localhost", "port": 9200}],
    use_ssl=False
)
print(client.info())
```

4.2.2 Task 2: Create Index

```
index_body = {
    "settings": {"index": {"knn": True}},
    "mappings": {
        "properties": {
            "content": {"type": "text"},
            "content_vector": {
                "type": "knn_vector",
                "dimension": 1024
            }
        }
    }
}
client.indices.create(index="documents", body=index_body)
```

4.2.3 Task 3: Index Documents

```
doc = {
    "content": "Sample document text",
    "content_vector": [0.1, 0.2, ...]  # 1024 dims
}
client.index(index="documents", body=doc)
```

4.2.4 Task 4: Implement Search

Implement hybrid search combining vector and BM25.

4.3 Repository

https://github.com/amornpan/Generic-RAG

4.4 Deliverables

Connection working

Index created

Documents indexed

Search implemented

Deadline: Sunday 23:59

5 Lab 5: Embeddings (3.75%)

5.1 Objectives

- Run embedding.py
- Understand chunking
- Add new documents

5.2 Tasks

5.2.1 Task 1: Study embedding.py

Review the pipeline:

1. Load documents from md_corpus/
2. Chunk documents
3. Create embeddings (bge-m3)
4. Index to OpenSearch

5.2.2 Task 2: Run Indexing

```
cd Generic-RAG
conda activate rag_env
python embedding.py
```

5.2.3 Task 3: Verify

```
curl http://localhost:9200/documents/_count
```

5.2.4 Task 4: Add New Documents

Add 3 new .md files to `md_corpus/` folder.

5.2.5 Task 5: Re-index

```
python embedding.py
```

5.2.6 Task 6: Test Search

Search for content from your new documents.

5.3 Repository

https://github.com/amornpan/Generic-RAG

5.4 Deliverables

Indexing completed

Documents verified

New documents added

Search tested

Deadline: Sunday 23:59

6 Lab 6: RAG + Local LLM + Streamlit (3.75%)

6.1 Objectives

- Setup Ollama
- Complete RAG pipeline
- Run Streamlit UI

6.2 Tasks

6.2.1 Task 1: Install Ollama

1. Download from https://ollama.ai
2. Install and run

6.2.2 Task 2: Pull Model

```
ollama pull qwen2.5:7b
```

6.2.3 Task 3: Test LLM

```
ollama run qwen2.5:7b
# Type: "What is RAG?"
```

6.2.4 Task 4: Run API with LLM

```
cd Generic-RAG
python api.py
```

6.2.5 Task 5: Run Streamlit

```
streamlit run app.py
```

6.2.6 Task 6: Test Complete Flow

1. Open http://localhost:8501
2. Ask questions
3. Verify RAG responses

6.2.7 Task 7: Modify Prompt

Edit the prompt template in api.py.

6.3 Repository

https://github.com/amornpan/Generic-RAG

6.4 Deliverables

Ollama running

LLM tested

Complete RAG working

Streamlit UI working

Deadline: Sunday 23:59

7 Lab 7: Docker Compose (3.75%)

7.1 Objectives

- Clone Advanced-RAG-Docker
- Run 6 services with Docker Compose
- Verify all services

7.2 Tasks

7.2.1 Task 1: Clone Repository

```
git clone https://github.com/amornpan/Advanced-RAG-Docker.git
cd Advanced-RAG-Docker
```

7.2.2 Task 2: Study docker-compose.yml

Review the 6 services:

Service	Port
opensearch_container	9200
embedding_container	-
search_api_container	8005
backend_container	8006
frontend_container	8501
ollama_container	11434

7.2.3 Task 3: Start Services

```
docker-compose up -d
```

7.2.4 Task 4: Check Status

```
docker-compose ps
```

7.2.5 Task 5: View Logs

```
docker-compose logs -f
```

7.2.6 Task 6: Test Application

Open http://localhost:8501

7.2.7 Task 7: Stop Services

```
docker-compose down
```

7.3 Repository

https://github.com/amornpan/Advanced-RAG-Docker

7.4 Deliverables

All 6 services running

Application accessible

Screenshots of docker ps

Document observations

Deadline: Sunday 23:59

8 Lab 8: CI/CD + Testing (3.75%)

8.1 Objectives

- Write pytest tests
- Create CI/CD pipeline
- Setup GitHub Actions

8.2 Tasks

8.2.1 Task 1: Write test_api.py

```
import pytest
from fastapi.testclient import TestClient
from api import app

client = TestClient(app)

def test_health():
    response = client.get("/health")
    assert response.status_code == 200
    assert response.json()["status"] == "healthy"

def test_search():
    response = client.post("/search",
        json={"query": "test", "top_k": 5})
    assert response.status_code == 200
```

8.2.2 Task 2: Run Tests

```
pytest tests/ -v
pytest tests/ --cov=. --cov-report=html
```

8.2.3 Task 3: Create Jenkinsfile

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps { sh 'docker-compose build' }
        }
        stage('Test') {
            steps { sh 'pytest tests/' }
        }
        stage('Deploy') {
            steps { sh 'docker-compose up -d' }
        }
    }
}
```

8.2.4 Task 4: Create GitHub Actions

Create `.github/workflows/ci.yml`:

```
name: CI/CD
on: [push, pull_request]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - run: docker-compose build
      - run: docker-compose up -d
      - run: sleep 60
      - run: curl -f http://localhost:8006/health
```

8.3 Repository

https://github.com/amornpan/Advanced-RAG-Docker

8.4 Deliverables

- Tests written
- Tests passing
- Jenkinsfile created
- GitHub Actions workflow
- Documentation

Deadline: Sunday 23:59

Summary: All Labs

Lab	Topic	Weight
1	Git + Python Fundamentals	3.75%
2	Docker + OpenSearch	3.75%
3	FastAPI	3.75%
4	OpenSearch Integration	3.75%
5	Embeddings	3.75%
6	RAG + Local LLM + Streamlit	3.75%
7	Docker Compose	3.75%
8	CI/CD + Testing	3.75%
	Total	**30%**

Repositories

- Generic-RAG: https://github.com/amornpan/Generic-RAG
- Advanced-RAG-Docker: https://github.com/amornpan/Advanced-RAG-Docker