Testing GenAI apps with Testcontainers and Docker

Garden State JUG
Jun 24, 2025

About me

Anna Chernyshova

Sr. Solutions Engineer @ Docker

Former Engineer @ AtomicJar (acquired by Docker),
the company behind Testcontainers

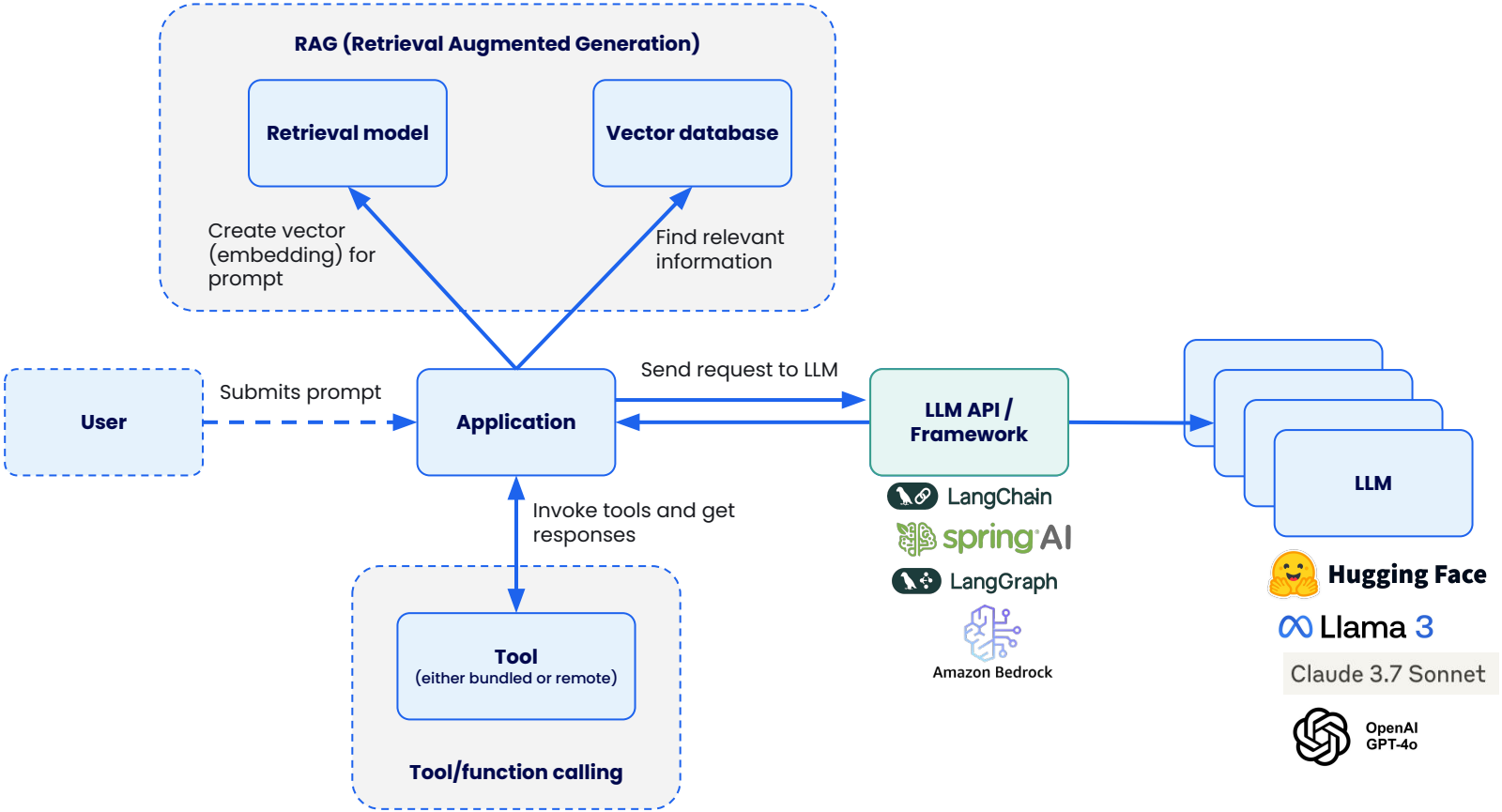
/chernyshovaanna

/AnnaChe_

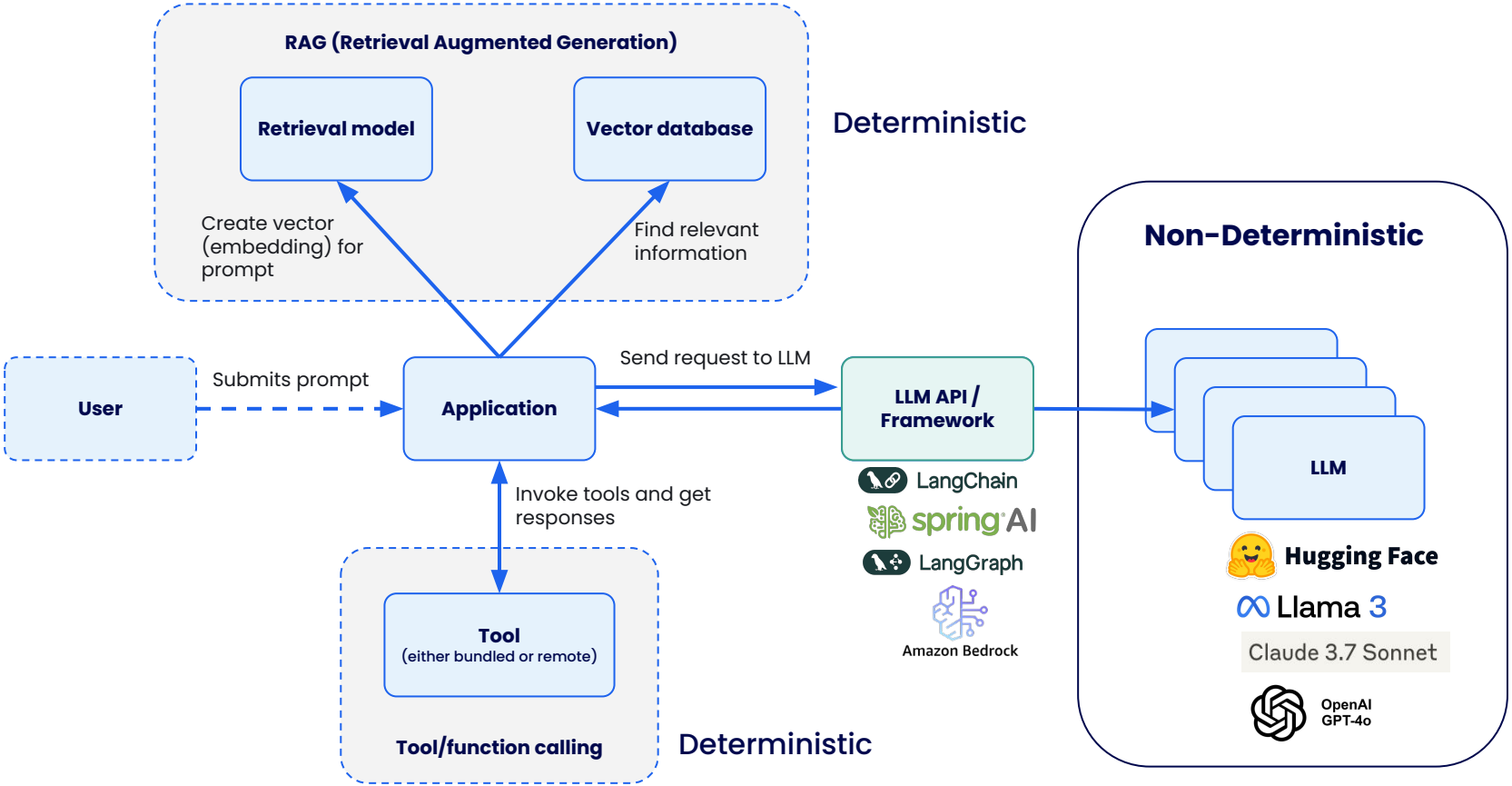
What we are going to see today:

01. GenAI in today's software
02. Testing of the deterministic part and tools for this
03. Testing of non-deterministic part and tools for this
04. Useful Links

The GenAI stack



Testing challenges



Testing of the deterministic components

Testing of the deterministic part is well established:

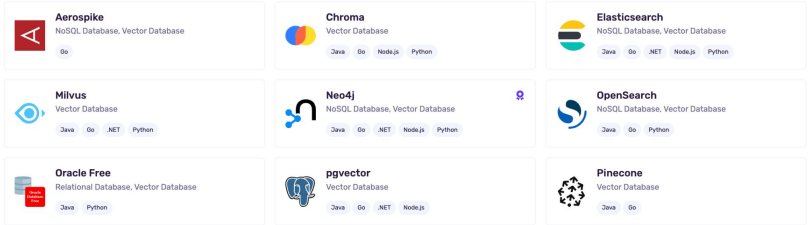
- Testing AWS service integrations using LocalStack
- Testing Spring Boot Kafka Listener using Testcontainers
- The simplest way to replace H2 with a real database for testing , etc..

Testing of the deterministic components

open source library for running your **app dependencies** (e.g. databases, message brokers, etc) as Docker **containers** without leaving your IDE.

```
var pgVector = new PostgreSQLContainer<>(DockerImageName.parse("pgvector/pgvector:pg16")
          .asCompatibleSubstituteFor("postgres"));
pgVector.start();
```

https://www.testcontainers.com/modules



Testcontainers Java: modules

JAR files providing access to the most used technologies:

- Relational DBs: Mysql, Postgres, ...
- **Vector DBs: Weaviate, Chroma, Qdrant, Milvus...**
- Non Relational DBs: Elasticsearch, Redis, MongoDB, Neo4j, Opensearch...
- Cloud Emulators: Localstack, Google Cloud, Azurite
- Keycloak, OpenFGA, Vault...
- ~80 different Java modules!
- Convenient API specific to each module.

https://www.testcontainers.com/modules

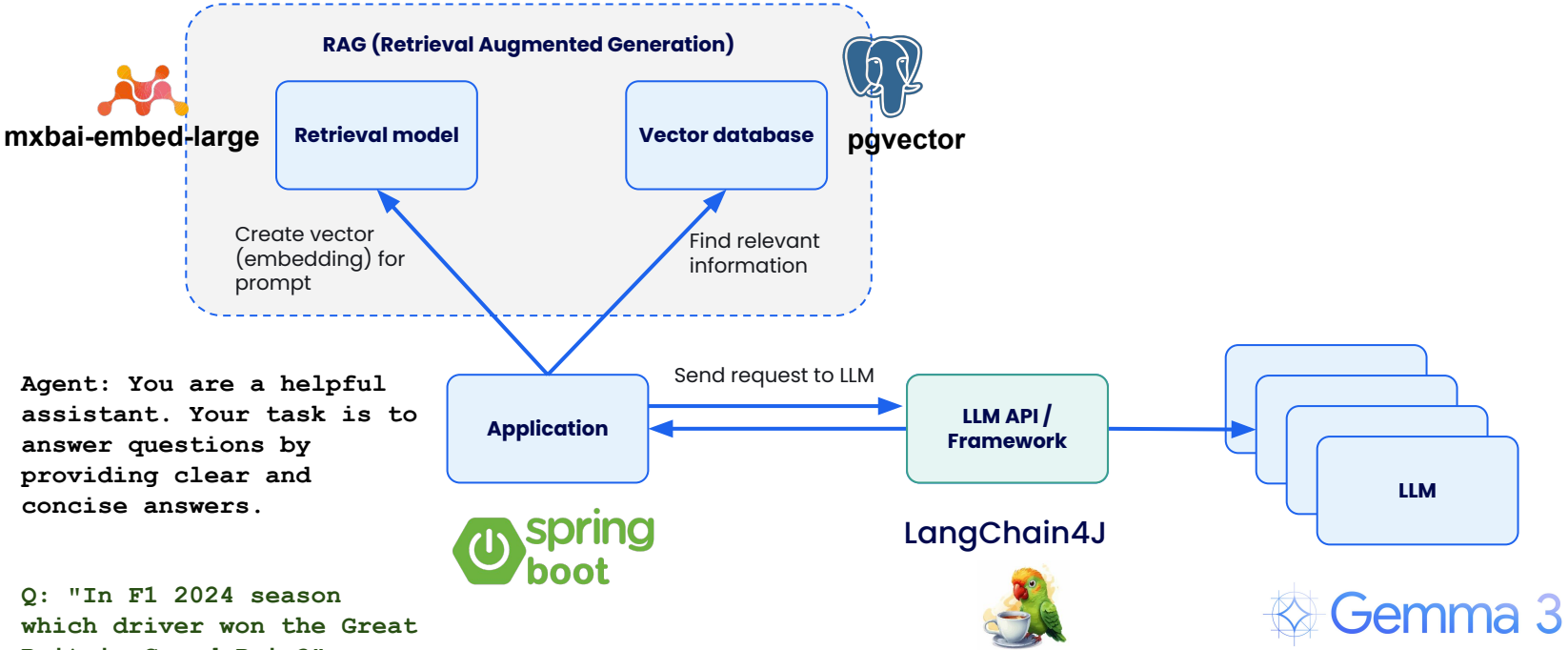
JAR: testcontainers-java

JAR: testcontainers-java/modules

Your Java app

How can we verify that non-deterministic LLM responses are correct?

The application



Langchain4j

Java implementation for Langchain: https://github.com/langchain4j/langchain4j

Community driven project, provides unified APIs to experiment with different LLMs or embedding stores (vector stores such as Pinecone or Milvus).

- ➔ **Switch between LLMs** without the need to rewrite your code (+15 providers: OpenAI, Anthropic, Google...)
- ➔ **Calculate embeddings** for words, texts, images... (+15 embedding models)
- ➔ **Talk to Vector databases** to look for similar/relevant documents to augment LLM responses (Retrieval Augmented Generation) (+20 vector databases)
    - ◆ Chroma, Milvus, pgVector, Pinecone, Qdrant, Weaviate...

langchain4j: embeddings

EmbeddingModel converts text into numerical vectors that capture meaning, enabling semantic search and similarity comparison.

Cosine similarity measures the angle between these arrows, not their length.

Visual Example:

- If two arrows point in the **same direction** → angle = 0° → cosine = 1 (identical)
- If two arrows point **perpendicular** → angle = 90° → cosine = 0 (unrelated)
- If two arrows point **opposite ways** → angle = 180° → cosine = -1 (opposite)

```
OpenAiEmbeddingModel model = OpenAiEmbeddingModel.builder()
            .apiKey(apiKey)
            .modelName("text-embedding-3-small”)
            .build();

Embedding catEmbedding = model.embed("A cat is a small
domesticated carnivorous mammal").content();

Embedding tigerEmbedding = model.embed("A tiger is a large
carnivorous feline mammal").content();

double similarity = CosineSimilarity.between(catEmbedding,
tigerEmbedding);
```

Cosine similarity between embeddings 0 and 1 is: 0.5904687602236495

langchain4j: RAG with AI Services

Question → Semantic Search → Context Retrieval → LLM Response Generation

Implementation with Langchain4J:

- Creates embedding model, vector store, and chat model instances
- Populates the embedding store with vectorized text segments
- Creates an assistant that uses both the chat model and retrieved context

Query Processing:

1. User questions trigger semantic search in embedding store
2. Top relevant context (maxResults=1) is retrieved using vector similarity
3. LLM generates response combining retrieved context with its knowledge

```
interface Assistant {
        String generate(String input);
}
public static void main(String[] args) {
        EmbeddingModel embeddingModel = buildEmbeddingModel();
        EmbeddingStore<TextSegment> store = buildEmbeddingStore();
        ingestion(embeddingModel, store);
        ChatLanguageModel chatModel = buildChatModel();

        Assistant assistant = AiServices.builder(Assistant.class)
                .chatLanguageModel(chatModel)
                .contentRetriever(
                        EmbeddingStoreContentRetriever.builder()
                                .embeddingModel(embeddingModel)
                                .embeddingStore(store)
                                .maxResults(1)
                                .build()
                        ).build();
        String response = assistant.generate("What is my favourite
        sport?");
…
}
```

Run models locally with Docker Model Runner

- Run LLMs locally
- Pull LLM models directly from Docker Hub
- Use the GPU to increase model performance
- Reduce # of separate tools to install/maintain

```
docker desktop enable model-runner --tcp 1234 (default)
```

Two scenarios to test

An application is talking to two agents:

Raw calls to the model

```
public static String getStraightAnswer() {
    ChatAgent straight = AiServices
            .builder(ChatAgent.class)
            .chatLanguageModel(chatModel())
            .build();
    return straight.chat(question);
}
```

Calls to the same model using RAG

```
public static String getRaggedAnswer() {
    ChatAgent ragged = AiServices
            .builder(ChatAgent.class)
            .chatLanguageModel(chatModel())
            .contentRetriever(contentRetriever()
            .build();
    return ragged.chat(question);
}
```

How to test #1: strings comparison

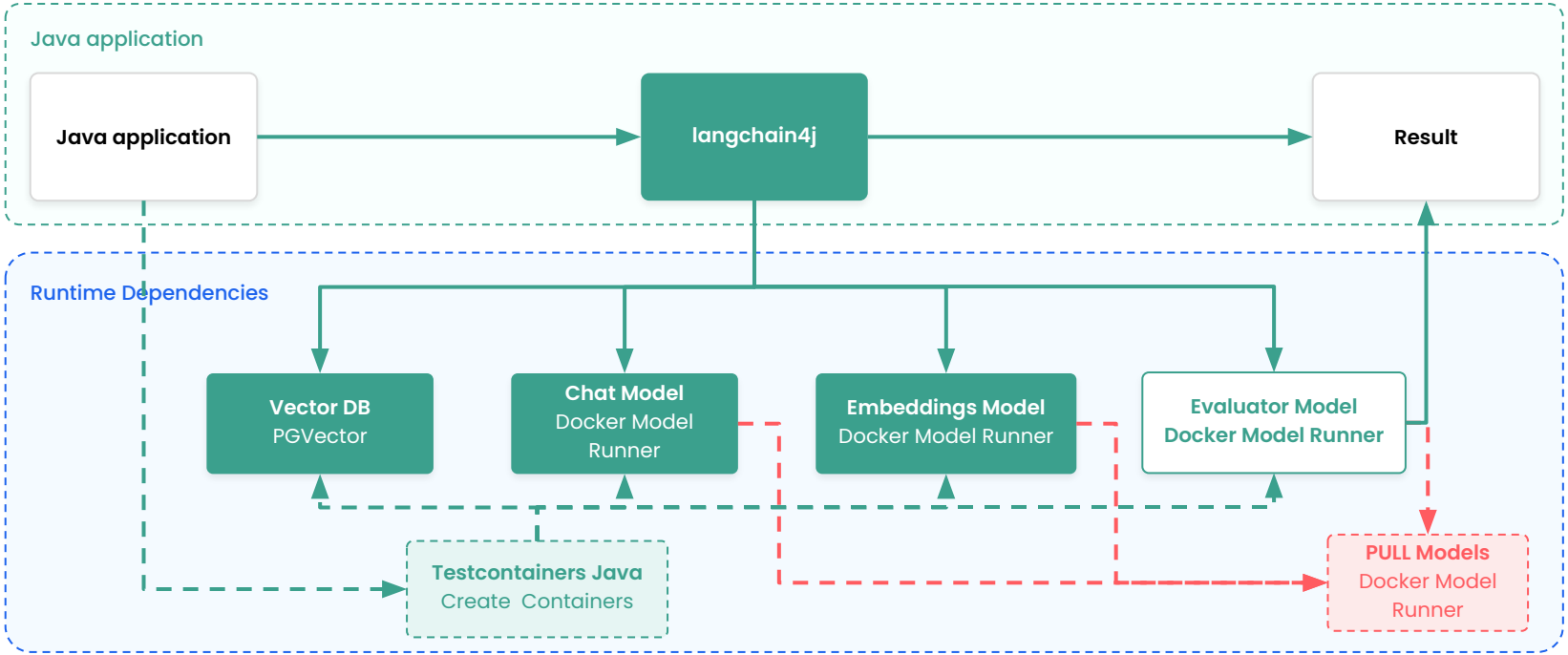
How to test # 2: cosine similarity

How to test #3: using an Evaluator

Enter Evaluators

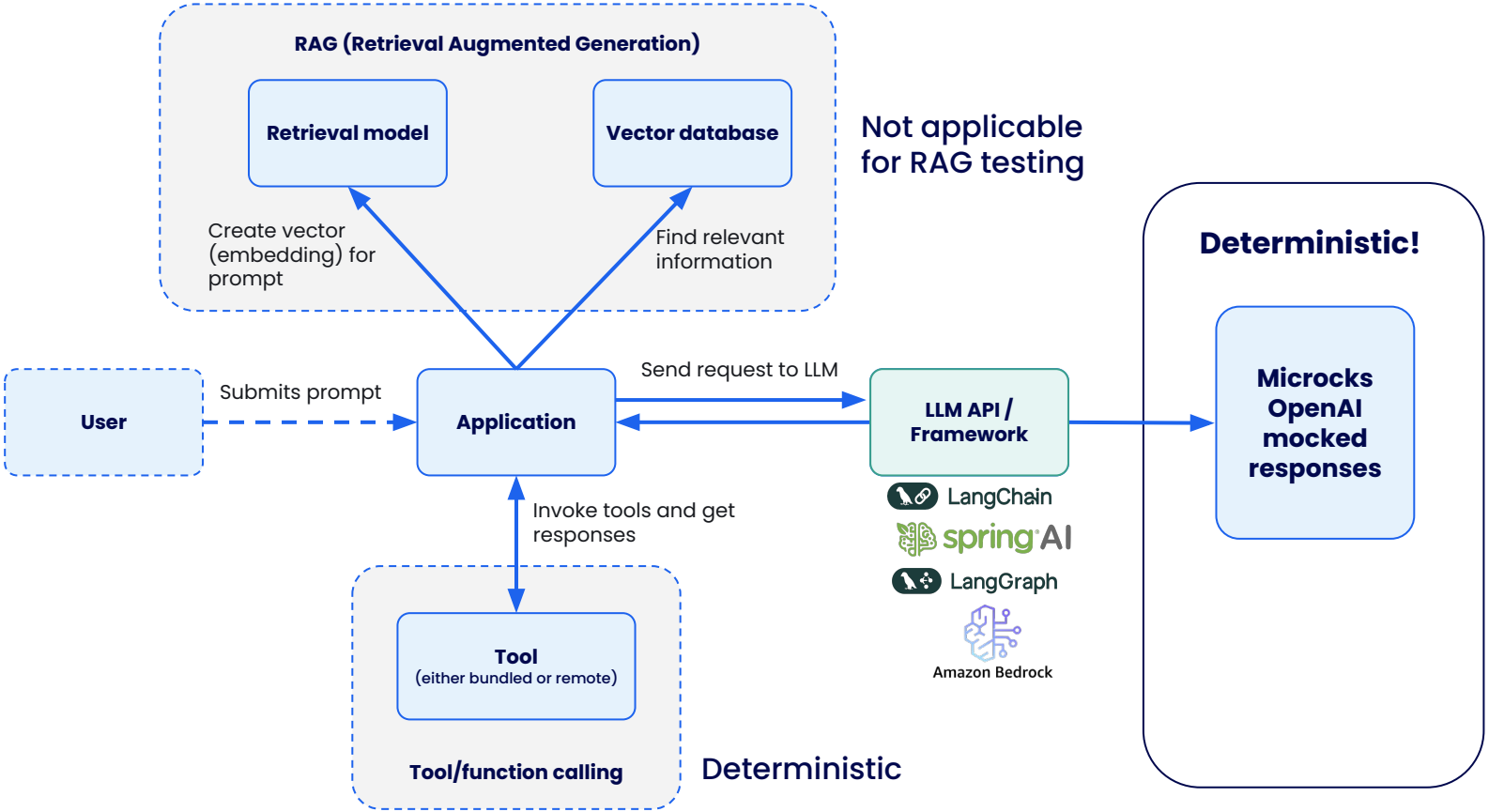
- AKA “LLM-as-a-Judge” (https://eugeneyan.com/writing/llm-evaluators/).
- Evaluate the quality of another LLM’s response to an instruction or query.
- Define a very **strict System Prompt:**
  - Provide **Instructions**: e.g. response format
  - Provide **reference examples**
- Define a very **strict User Prompt:**
  - Provide a **detailed format**: *### question ### answer ### reference ###*.
  - Provide a **reference** (e.g. in the test as an expectation)
  - Structured output, **semantic/style constraints**
    - Respond with “yes” or “no” including the reasoning.

Adding an Evaluator



How to test negative path scenarios?

Testing with the Mocked Model



Testcontainers Microcks module

Model's API is fragile — requires extensive negative path testing

Test with a mock:

- API returns semantically incorrect answer
- API is too slow
- API returns complete nonsense
- API returns incorrect or slightly incorrect schema
- …

MICROCKS

```
@Container
static MicrocksContainer microcks =
new MicrocksContainer
("quay.io/microcks/microcks-uber:latest")
.withMainArtifacts("openai-api-mock.yaml");
```

Simple example:

How to make sure that the Validator Agent is evaluating LLM response properly?

openai-api-mock.yaml

```
[.....]
x-microcks-operation:
 dispatcher: SCRIPT
[.....]
examples:
  hallucinate:
    summary: Hallucinate response
    value:
      id: "chatcmpl-123"
      object: "chat.completion"
      created: 1677652288
      model: "hallucinate"
      choices:
        - index: 0
          message:
            role: "assistant"
            content: "Lewis Hamilton won the 2024 Galaxy GP as part of the USS
Enterprise team."
          finish_reason: "stop"
      usage:
        prompt_tokens: 12
        completion_tokens: 20
        total_tokens: 32
[.....]
```

Useful links

Application example

https://github.com/DockerSolutionsEngineering/generative-ai-with-testcontainers

Docker Model Runner

docker model push needs* **gguf

https://docs.docker.com/model-runner/

https://github.com/docker/hello-genai

Testcontainers

https://github.com/testcontainers/testcontainers-java

https://github.com/testcontainers/workshop