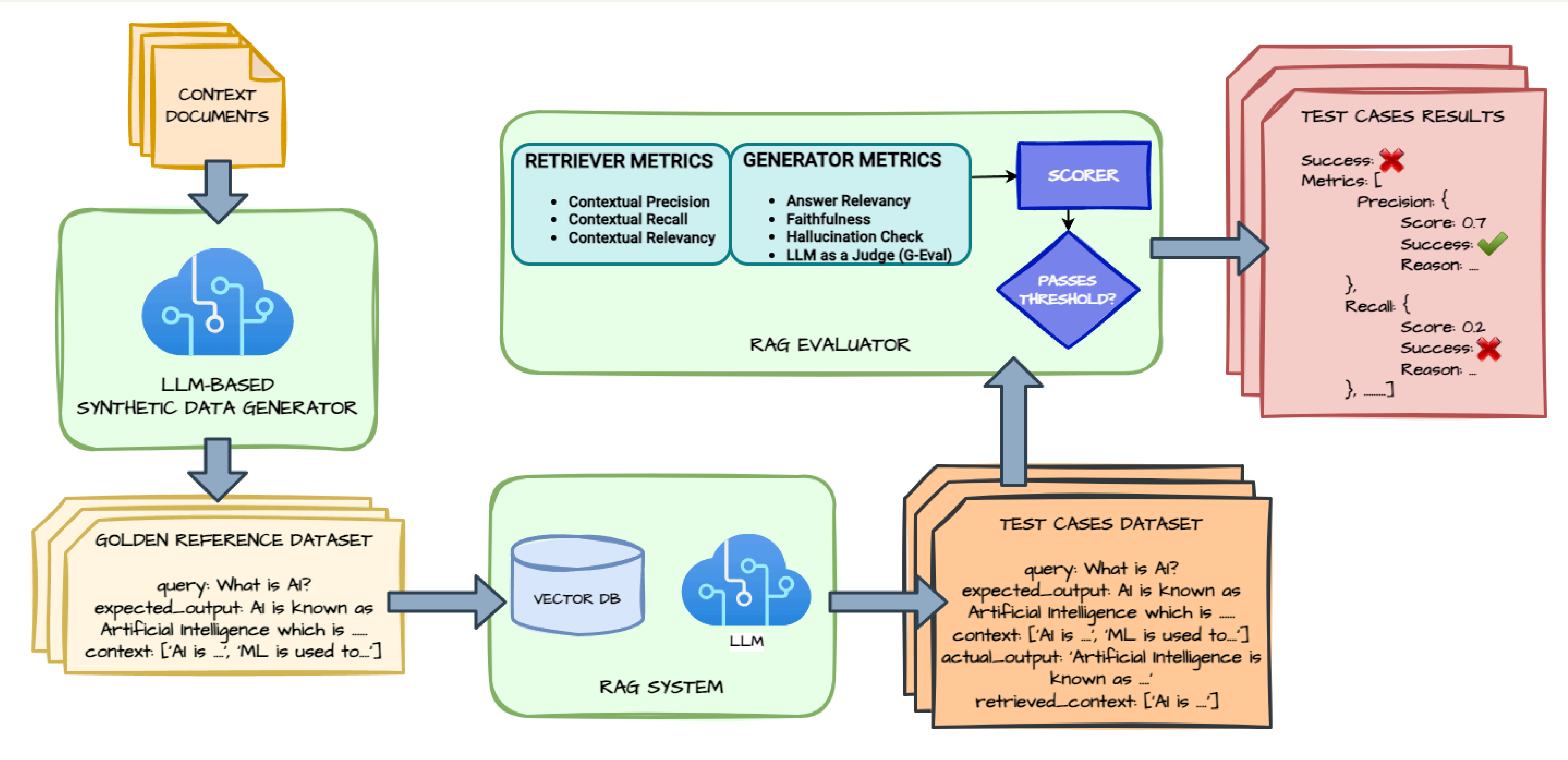
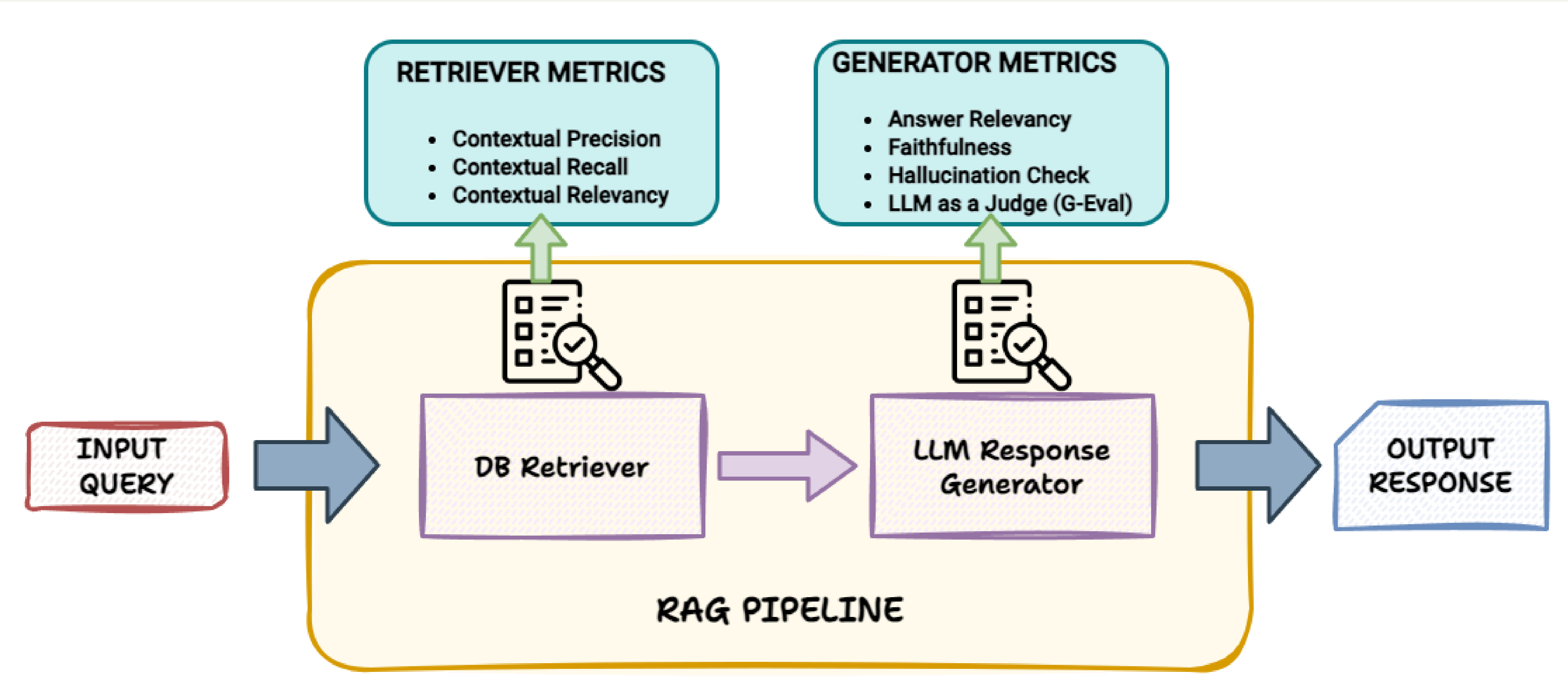
Guide to End-to-End RAG Systems Evaluation



Dipanjan (DJ)

Standard RAG System Evaluation Metrics



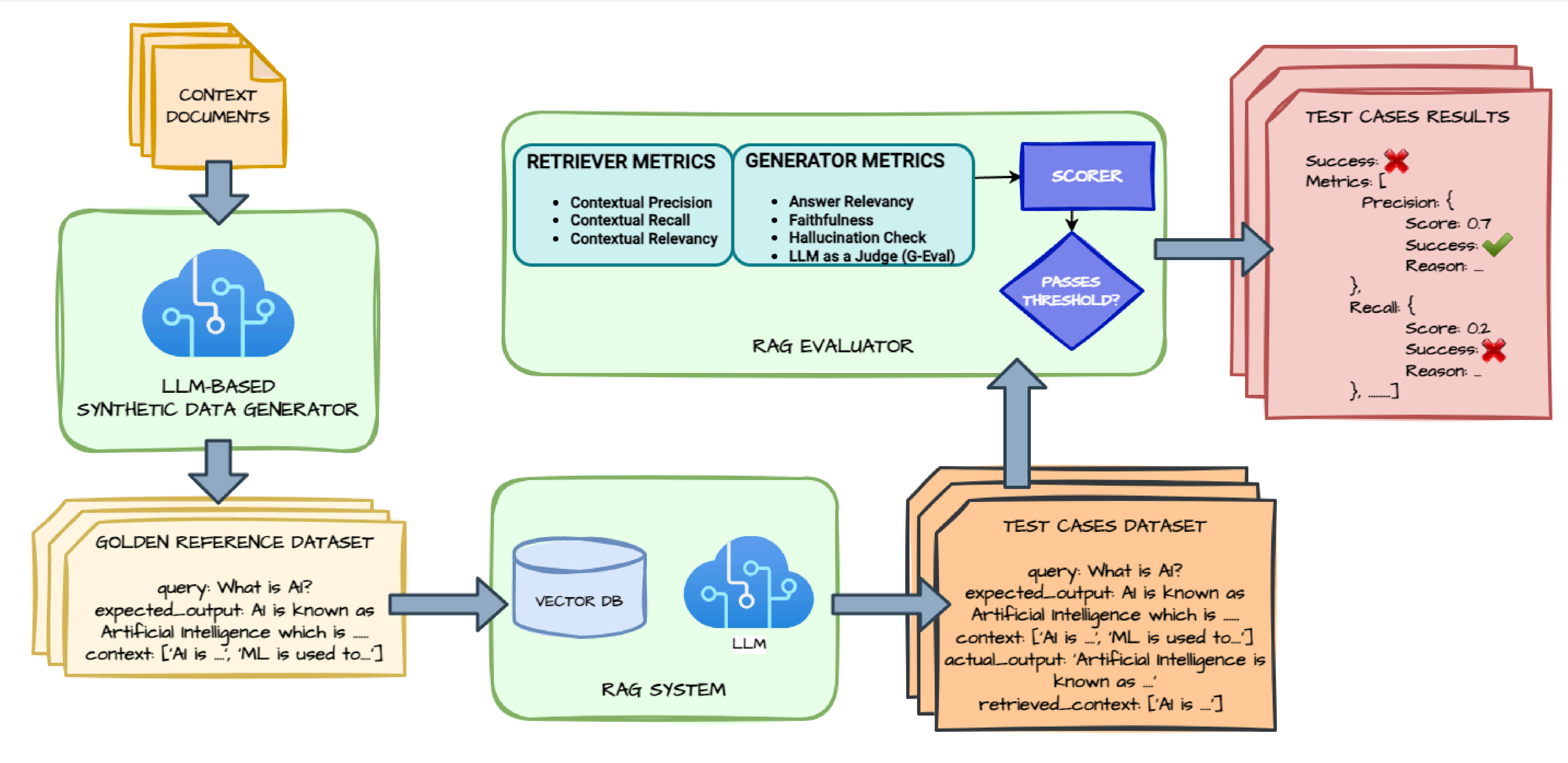
Two major points in a RAG System need evaluation

- **Retriever:** This is where we measure retrieval performance from the Vector DB for input queries
  - **Contextual Precision:** Relevant retrieved context to input query should rank higher
  - **Contextual Recall:** Retrieved context should align with expected ground truth response
  - **Contextual Relevancy:** Relevancy of statements in retrieved context to the input query should be more in count

- **Generator:** This is where we measure the quality of generated responses from the LLM for input queries and retrieved context
  - **Answer Relevancy:** Relevancy of statements in generated response to the input query should be more in count or semantically similar (LLM-based or semantic similarity)
  - **Faithfulness:** Count of truthful claims made in the generated response w.r.t the retrieved context should be more
  - **Hallucination Check:** Number of statements in generated response which contradict the ground truth context should be minimal
  - **Custom LLM as a Judge:** You can create your own judging metrics based on custom evaluation criteria as needed

Source: Dipanjan (DJ)

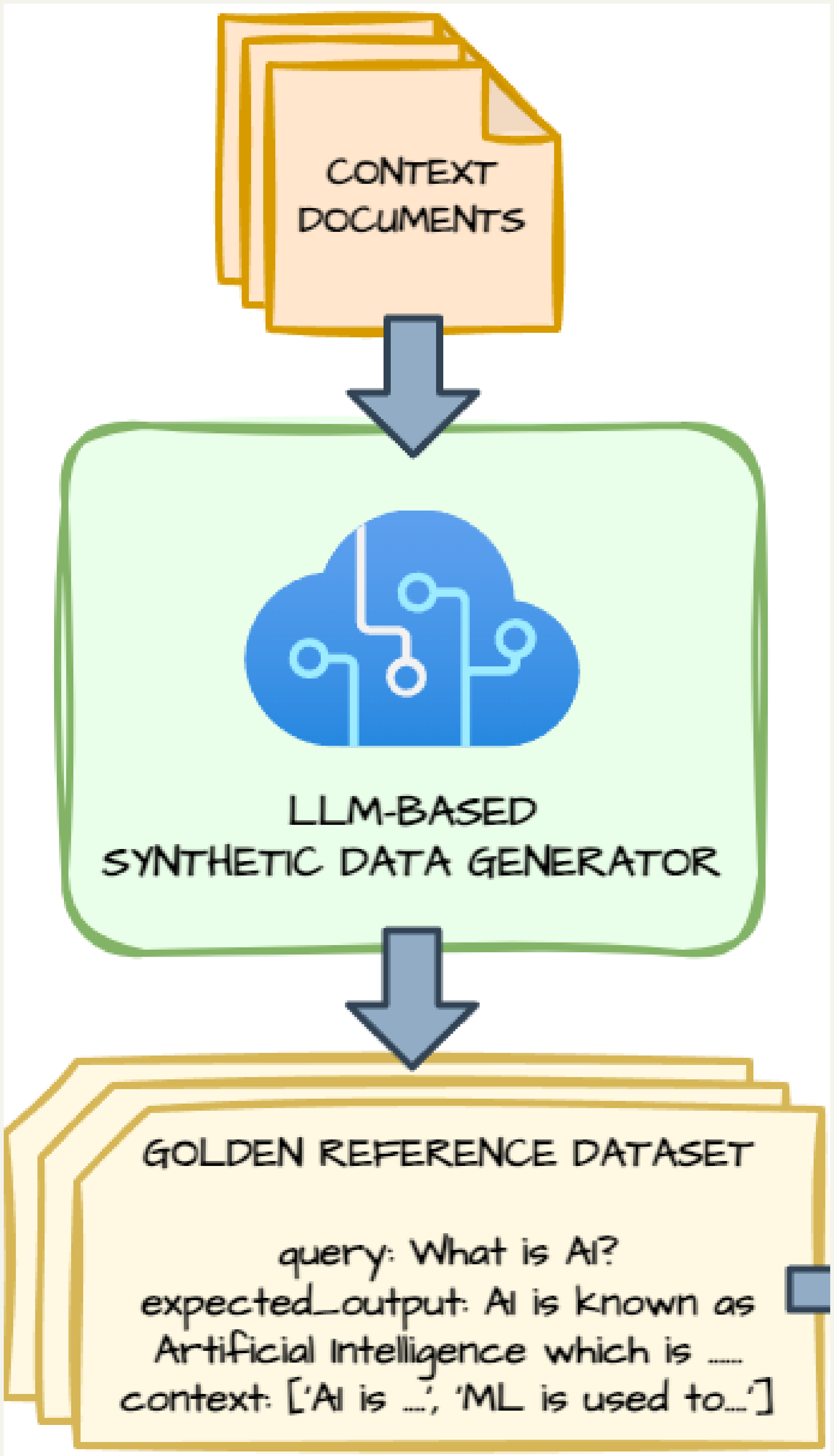
End-to-End RAG Evaluation Workflow



- The following key steps are necessary to enable end-to-end evaluation of a RAG System
  - Build a RAG System which can return generated responses and retrieved context sources in one go
  - Using your context documents generate golden reference data samples using LLMs or manually
  - Run input queries from each reference sample through your RAG System and get generated responses
  - Create Test Cases using your golden reference data and actual generated responses and retrieved contexts
  - Use any standard RAG Evaluation framework to evaluate based on metrics and settings of your choice
  - Review performance of your system based on results and iterate

Source: Dipanjan (DJ)

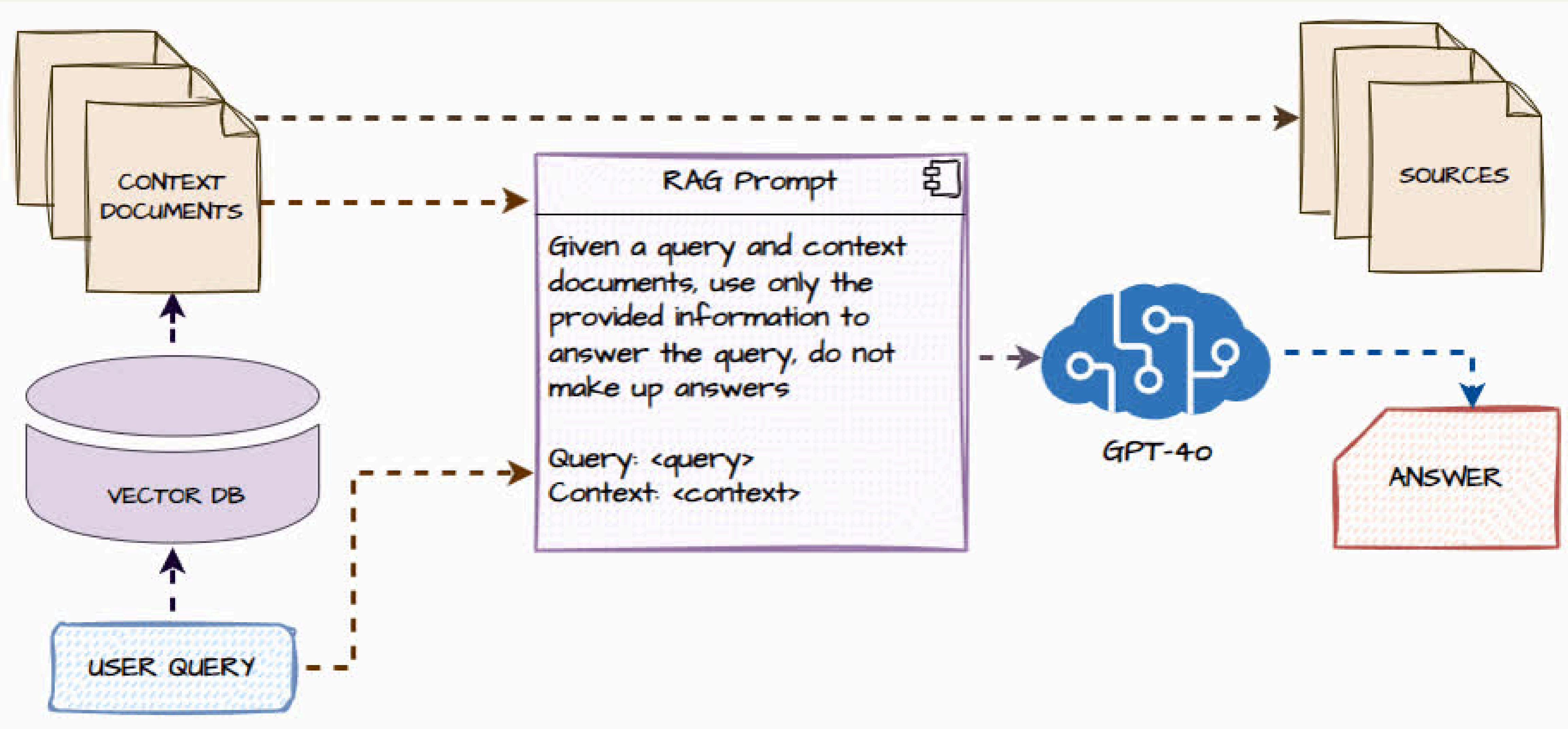
LLM-based Sythetic Golden Reference Data Generator



- **Create Golden Reference Data samples manually or using an LLM synthetically**
- **Golden reference data samples would consist of the following:**
  - **Input Query:** Input question to the RAG system
  - **Expected Output:** Ground truth answer to be expected from the LLM Generator
  - **Context:** Expected ground truth context which should be retrieved

Source: https://www.anthropic.com/news/contextual-retrieval

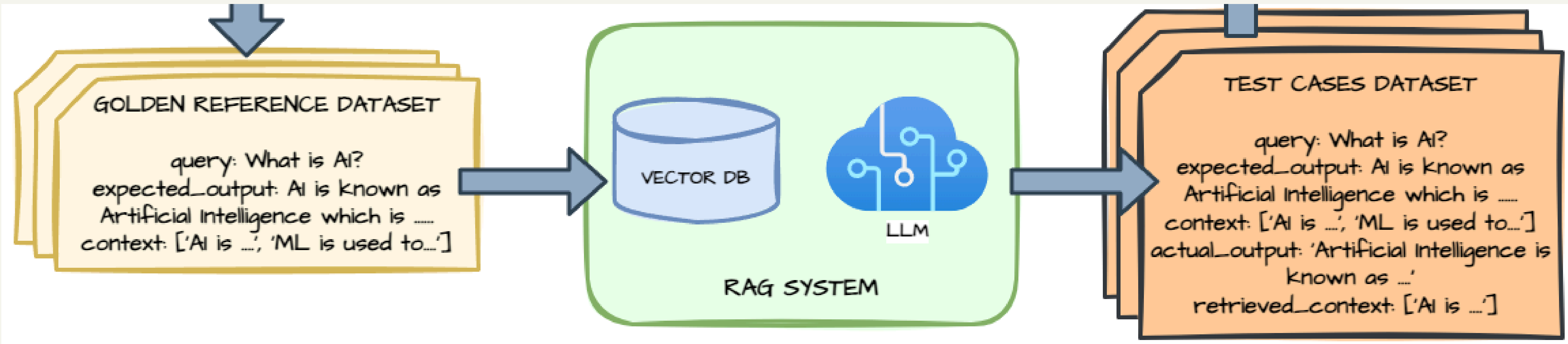
RAG System with Sources



- **Build a RAG System as usual which can return the generated response to any input query**
- **Besides the response also return the retrived source context**
- **This helps in evaluating retriever and generator metrics in one go**
- **Avoids having to run separate queries on Vector DB for evaluating retriever metrics and RAG System for generator metrics for each reference data sample**

Source: Dipanjan (DJ)

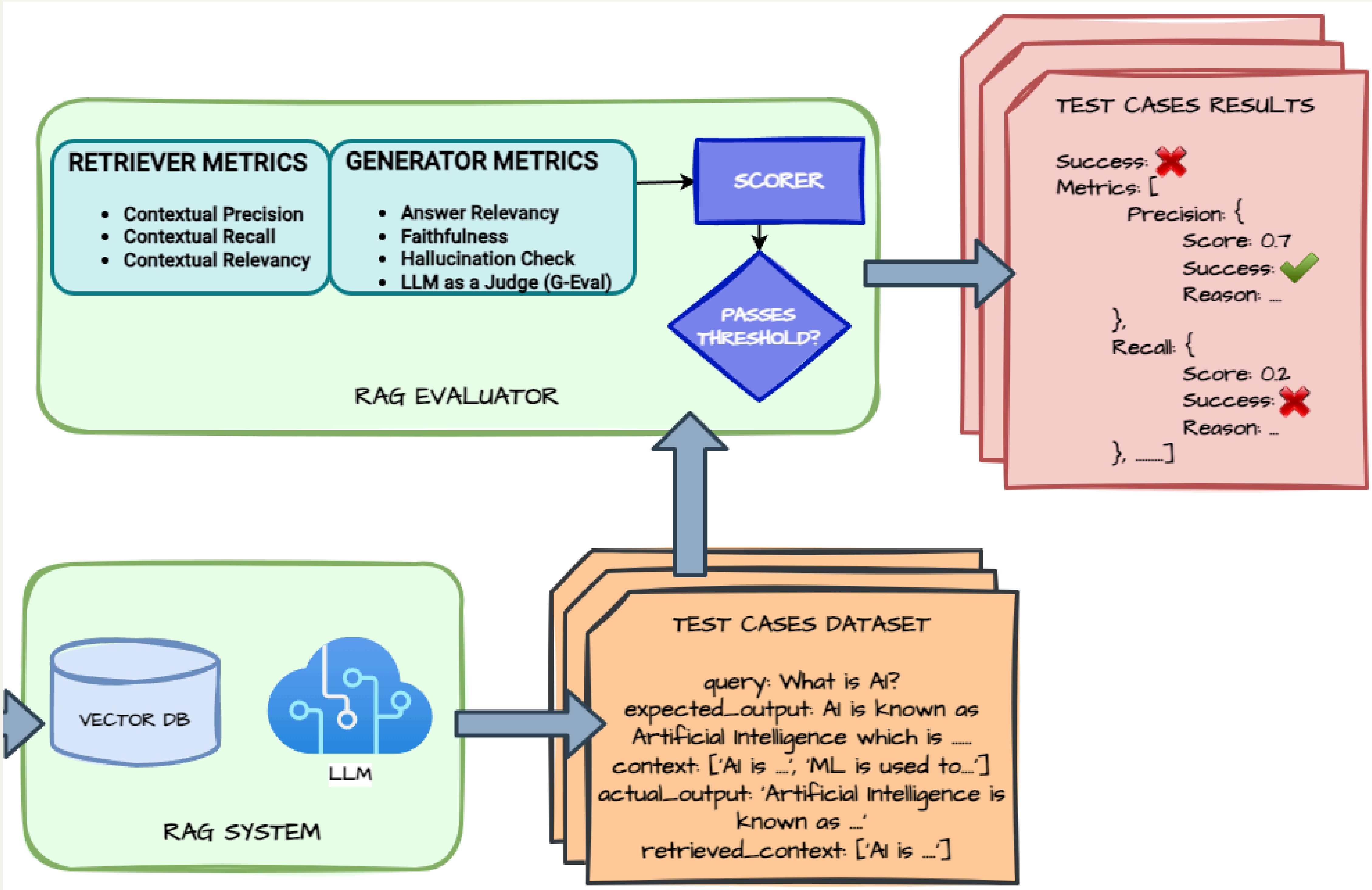
Create Evaluation Test Cases



- **Here we take the input query of each golden reference data sample**
- **Pass the query to our RAG System and take the Retrieved Context and LLM Response as output**
- **Append them to each golden reference data sample to create a test case**
- **Each Test Case Sample will consist of the following:**
  - **Input Query:** Input question to the RAG system
  - **Expected Output:** Ground truth answer to be expected from the LLM Generator
  - **Context:** Expected ground truth context which should be retrieved
  - **Actual Output:** The actual response from the RAG System's LLM Generator
  - **Retrieved Context:** The actual retrieved context from the RAG System's Vector DB Generator

Source: Dipanjan (DJ)

Run RAG Evaluation on Test Cases



- **Define the RAG Metrics you want to evaluate each test case on in terms of:**
  - Metric Definition
  - Pass or Fail Threshold
  - Specific evaluation instructions in case of custom metrics
- **Evaluate each test case and store the metrics**
- **Visualize on your dashboard as needed and improve system over time**

Source: Dipanjan (DJ)

RAG Evaluation Example with DeepEval

```
from deepeval import evaluate
from deepeval.metrics import ContextualPrecisionMetric, ContextualRecallMetric, ContextualRelevancyMetric
from deepeval.metrics import AnswerRelevancyMetric, FaithfulnessMetric, HallucinationMetric
from deepeval.metrics.ragas import RAGASAnswerRelevancyMetric

eval_dataset.test_cases = [....] # create your test cases
contextual_precision = ContextualPrecisionMetric(threshold=0.5, include_reason=True, model="gpt-4o")
contextual_recall = ContextualRecallMetric(threshold=0.5, include_reason=True, model="gpt-4o")
contextual_relevancy = ContextualRelevancyMetric(threshold=0.5, include_reason=True, model="gpt-4o")
answer_relevancy = AnswerRelevancyMetric(threshold=0.5, include_reason=True, model="gpt-4o")
faithfulness = FaithfulnessMetric(threshold=0.5, include_reason=True, model="gpt-4o")
hallucination = HallucinationMetric(threshold=0.5, include_reason=True, model="gpt-4o")
ragas_answer_relevancy = RAGASAnswerRelevancyMetric(threshold=0.5, embeddings=OpenAIEmbeddings(),
                                                    model="gpt-4o")

eval_results = evaluate(test_cases=eval_dataset.test_cases,
                        metrics=[contextual_precision, contextual_recall, contextual_relevancy,
                                 answer_relevancy, ragas_answer_relevancy, faithfulness, hallucination])


## EVAL OUTPUT ##

Evaluating 10 test case(s) in parallel: |██████████|100% (10/10) [Time Taken: 00:39,  3.98s/test case]
======================================================================

Metrics Summary

  - ✅ Contextual Precision (score: 1.0, threshold: 0.5, strict: False, ....)
  - ❌ Contextual Recall (score: 0.25, threshold: 0.5, strict: False, ....)
  - ❌ Contextual Relevancy (score: 0.3333333333333333, threshold: 0.5, strict: False, ....)
  - ✅ Answer Relevancy (score: 1.0, threshold: 0.5, strict: False, ....)
  - ❌ Answer Relevancy (ragas) (score: 0.0, threshold: 0.5, strict: False, ....)
  - ✅ Faithfulness (score: 1.0, threshold: 0.5, strict: False, ....)
  - ❌ Hallucination (score: 1.0, threshold: 0.5, strict: False, ....)
```

- **You can leverage libraries like DeepEval and Ragas to make things easier for you or even create your own custom eval metrics**

Source: Dipanjan (DJ)