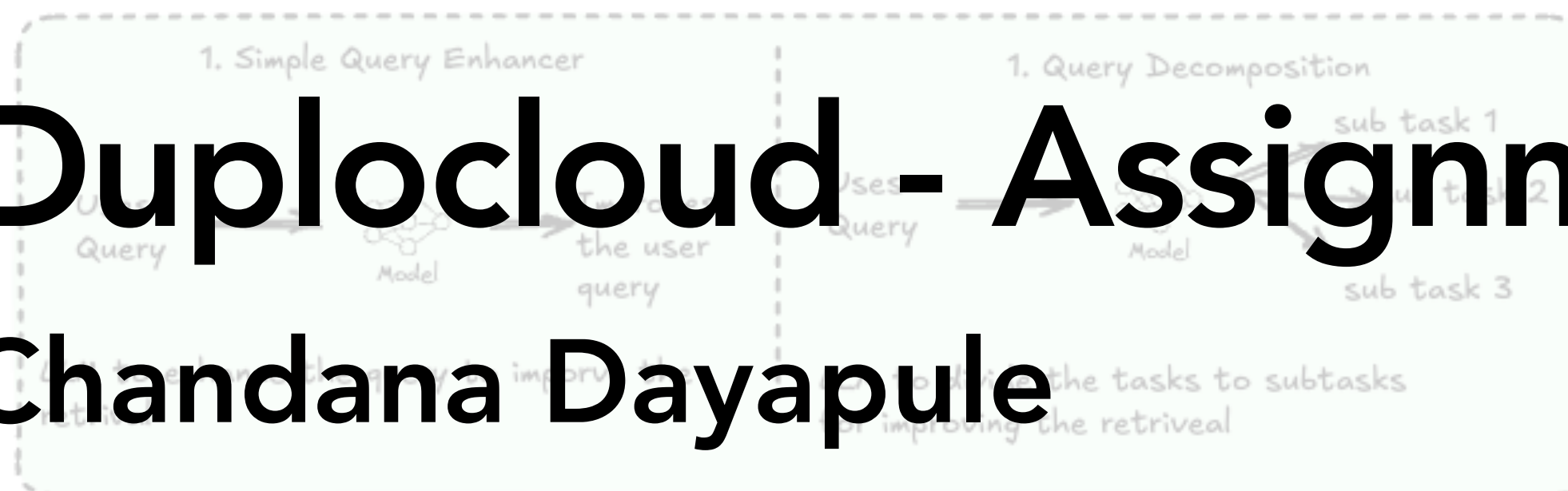
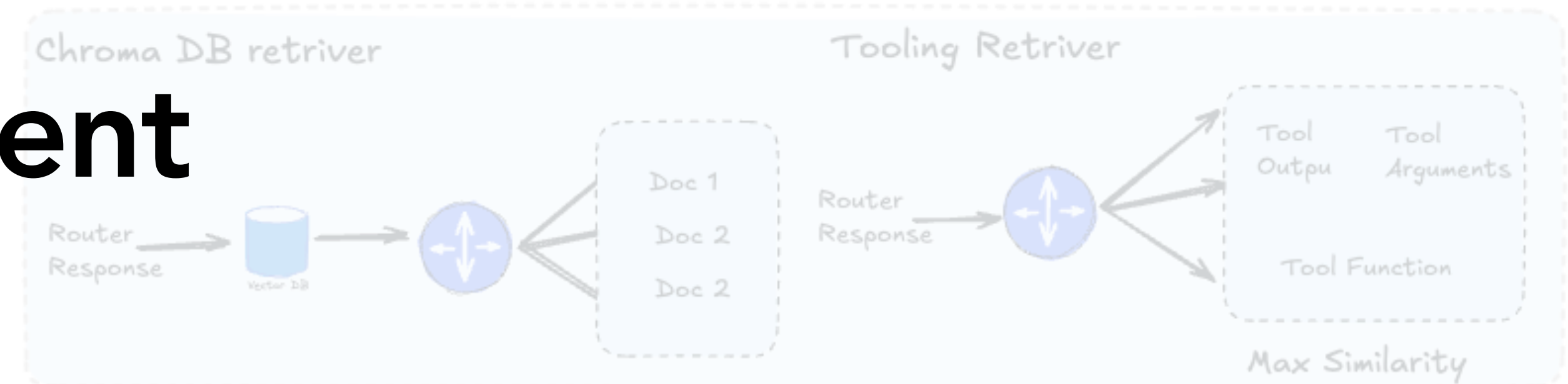


Overall Workflow

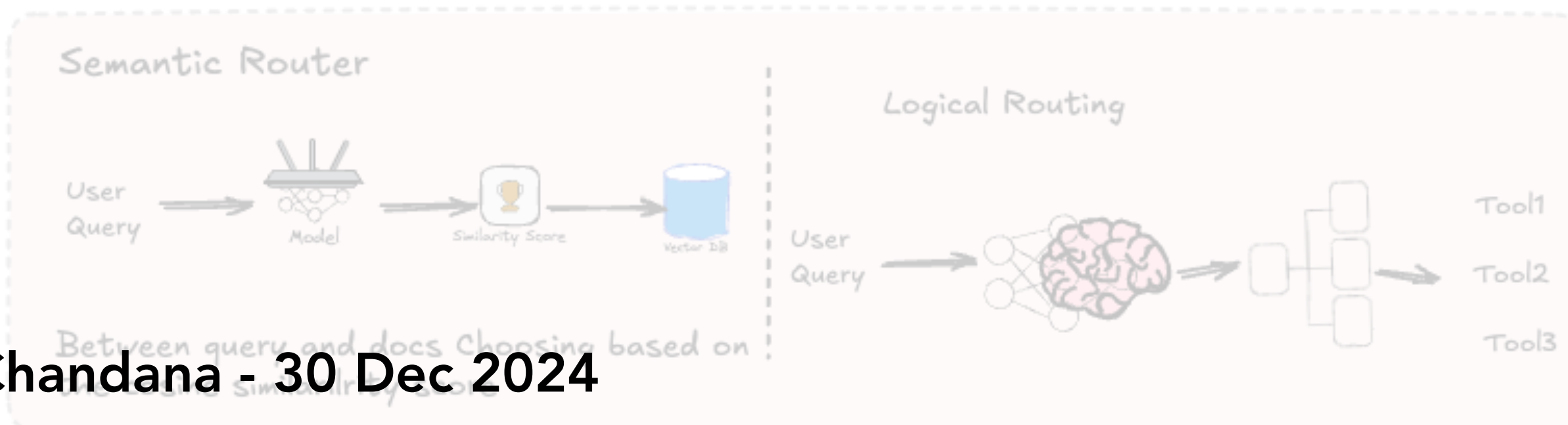
Translator



Retriver



Router



Generator

Duplocloud - Assignment

Chandana Dayapule

Chandana - 30 Dec 2024

Content

Part-A

1. Overview
2. Workflow
3. Modules built
4. Translator
5. Router
6. Retrriver
7. Generator
8. Data Indexing
9. Github crawler
10. System Architecture

Part-B

1. Test Cases
 1. Duploccloud Queries
 2. Non-Duploccloud Queries
 3. Web Search Queries
 4. Real time data Queries
2. Assumptions

Part-C

1. Future Work
 1. VectorDB
 2. Data
 3. RAG Pipeline
 1. Translator
 2. Decomposer
 3. Retriever
 4. Generator
 5. Indexer
 6. Testing

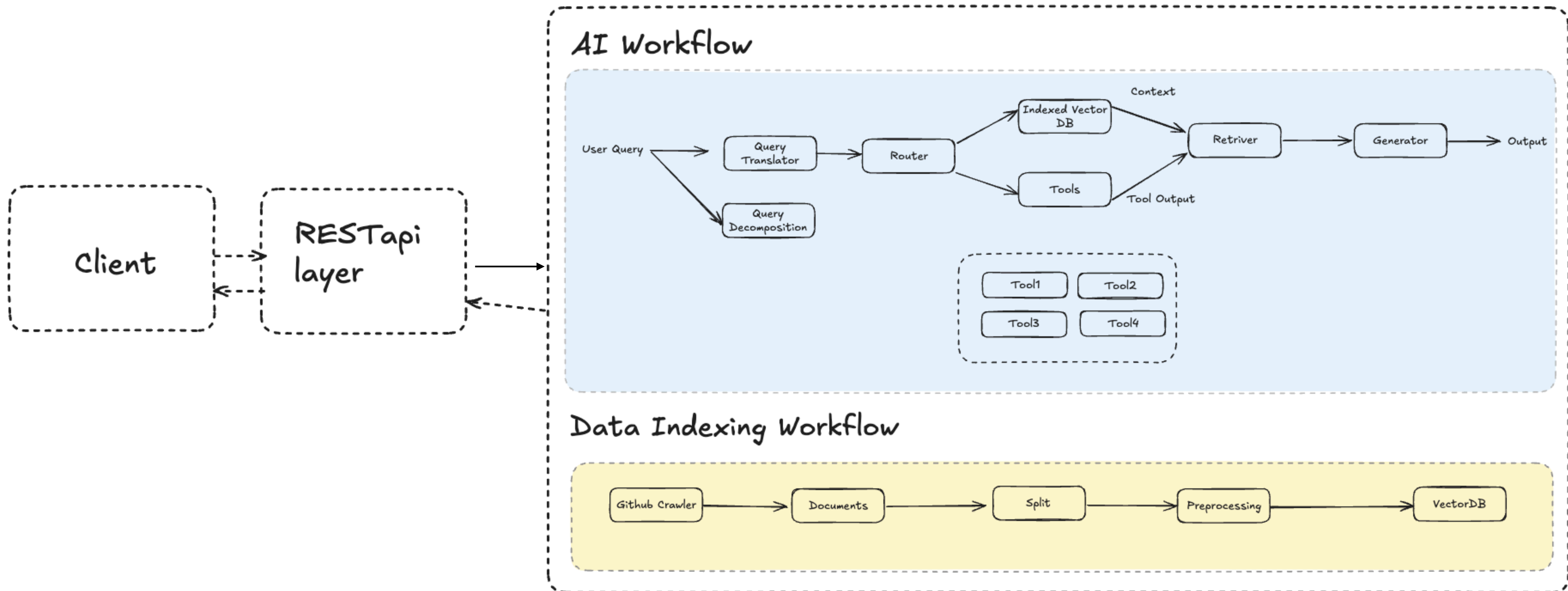
Assignment Problem Overview

1. RAG - DuploCloud AI agent
2. Web search functionality
3. Exposing to Backend API
4. Dockerization

Additional Requirements

1. Source code
2. README file
3. API Documentation

Architectural Diagram



Code Implementation Modules

Modules Built

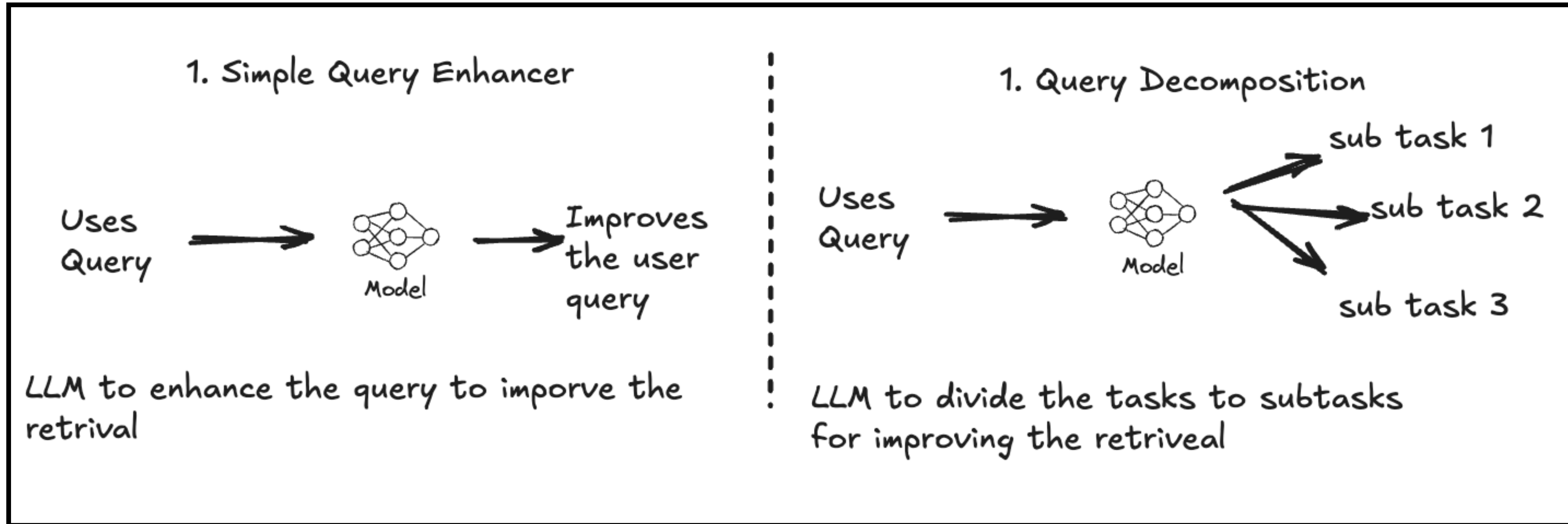
1. Query Translator
2. Query Router
3. Query Retriever
4. Query Generator

Vector DB indexing

1. Github crawler
2. Docs Loader
3. Docs splitter or chunking
4. Indexing Vector db

1. Query Translator

1. Query Translator



Method 1 - Simple Query Enhancer

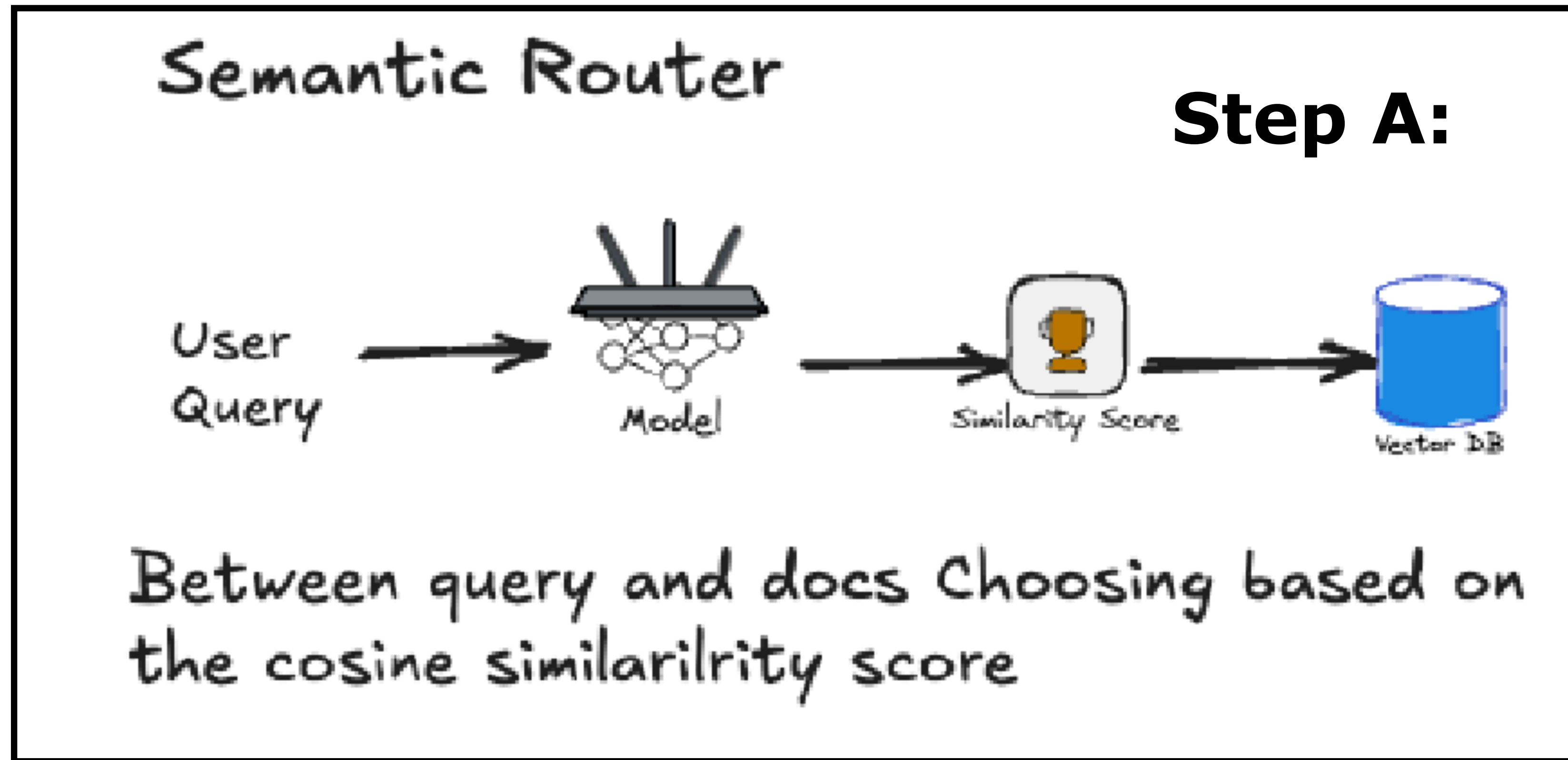
Translated into a format that matches the retrieval mechanism.

Method 2 - Query Decomposition

Splitting a complex query into simpler, more manageable sub-queries.

2.1 Query Router - Semantic Router

Query router is responsible for directing queries to the most appropriate data source



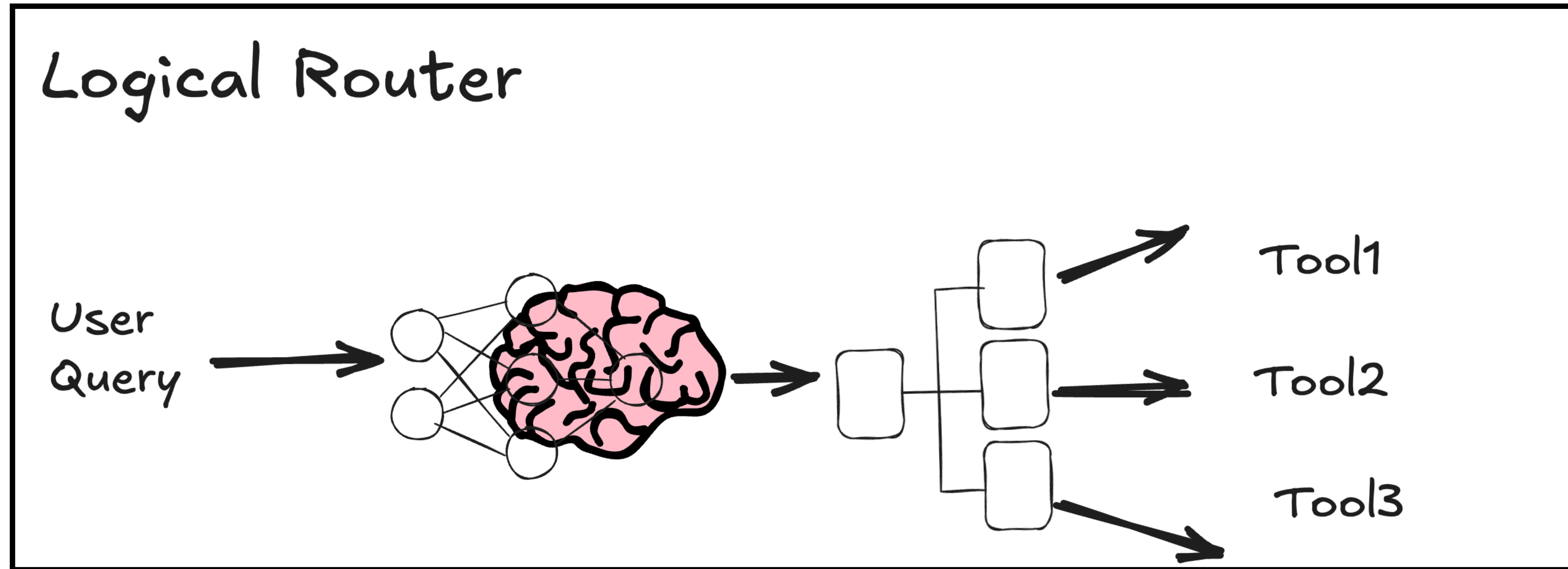
This task is performed in three steps

1. Analysis
2. Decision making
3. Execution

This module takes an user query, and makes a decision what type (Vector DB, Tool) of query the user requested, The current classification include,

1. Duplocloud Github documents
2. Wikipedia search
3. Brave search

2.2. Query Router - Tooling Router



Based on the logical flow Of the Tooling module from The description the LLM Logically routes to one

4. Query Retrival

Querying a

1. Vector database or
2. Tools to

For Vectoredb - retrieve the most relevant top k documents based on the semantic similarity to a query.

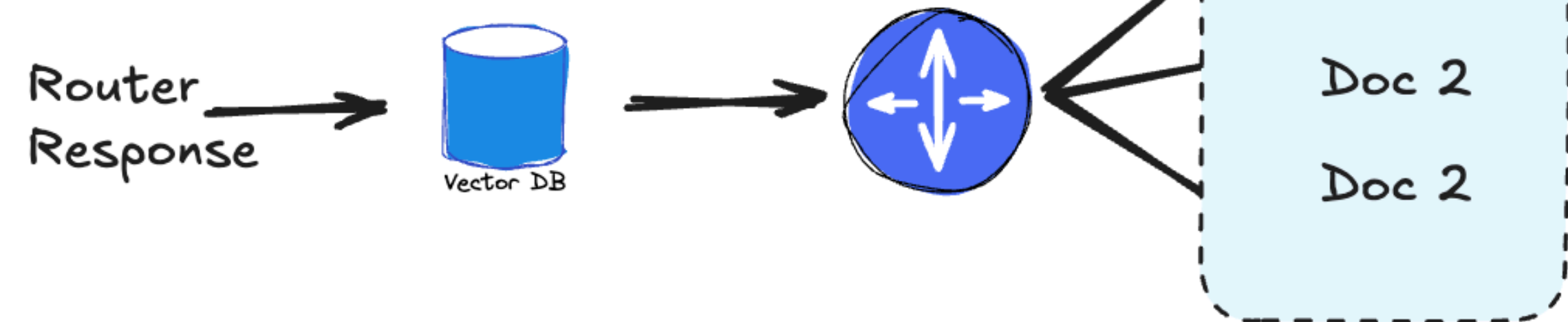
For Tooling - using LLM to retrieve the context

This module takes an user query, and makes a decision what type (Vector DB, Tool) of query the user requested,

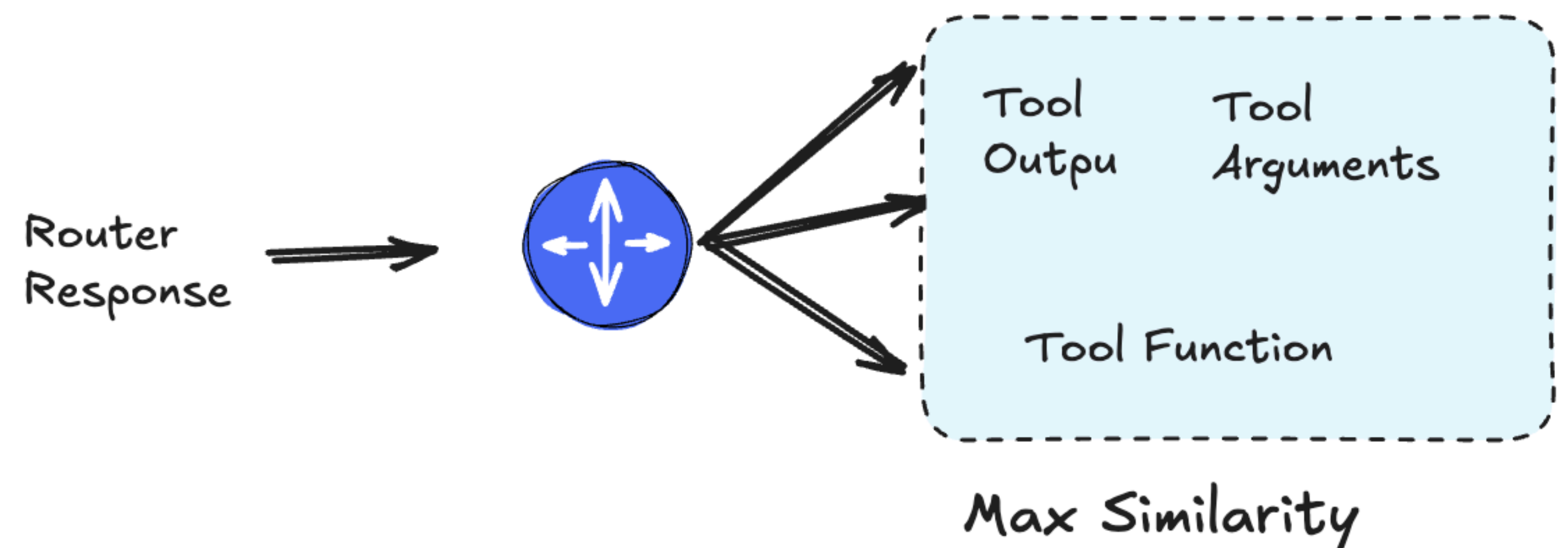
The current classification include,

1. Duplocloud Github documents
2. Wikipedia search
3. Brave search

Chroma DB retriver



Tooling Retrival



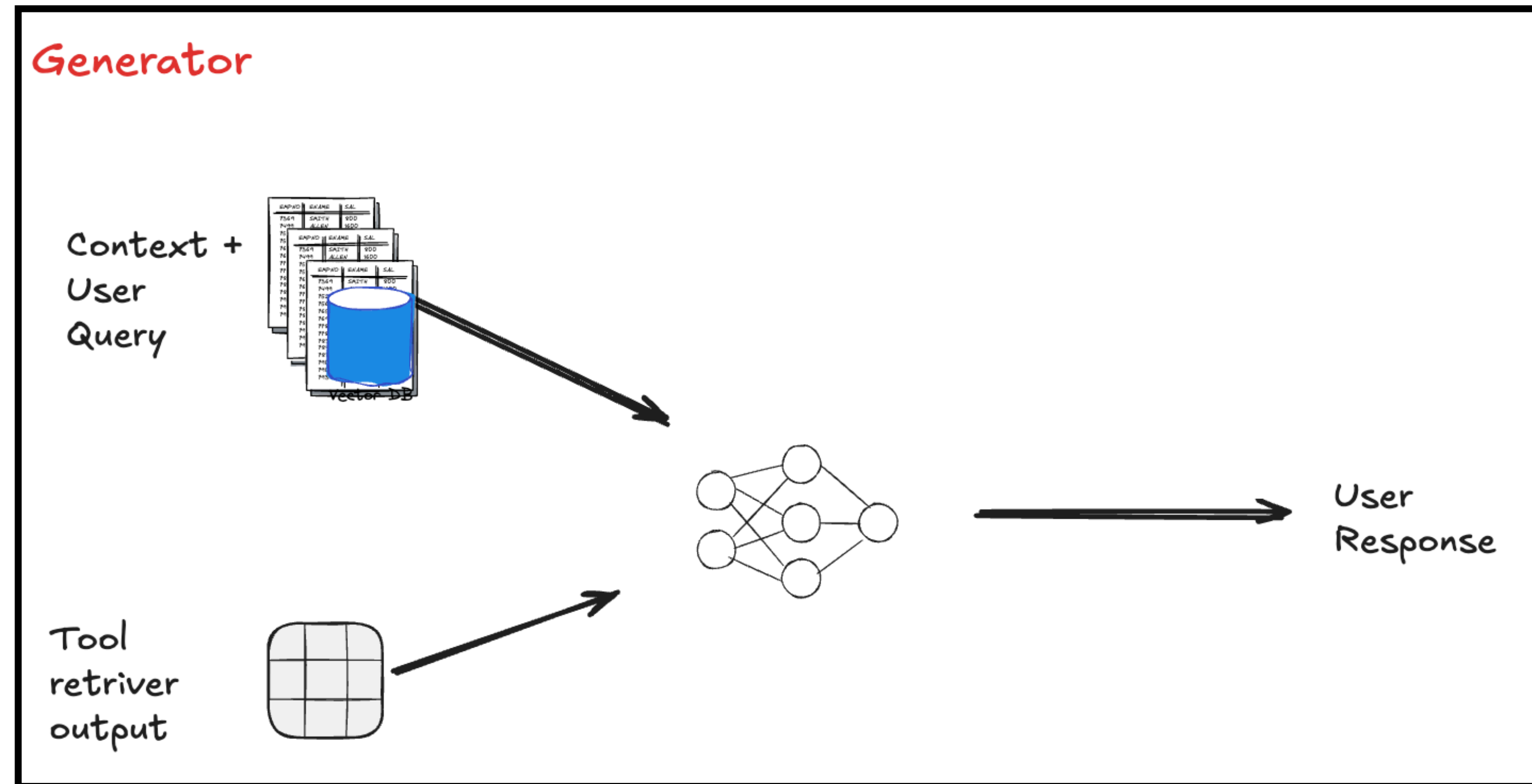
5. Generator

Generates answer based on input user query + context if from vector db or tooling output to the llm

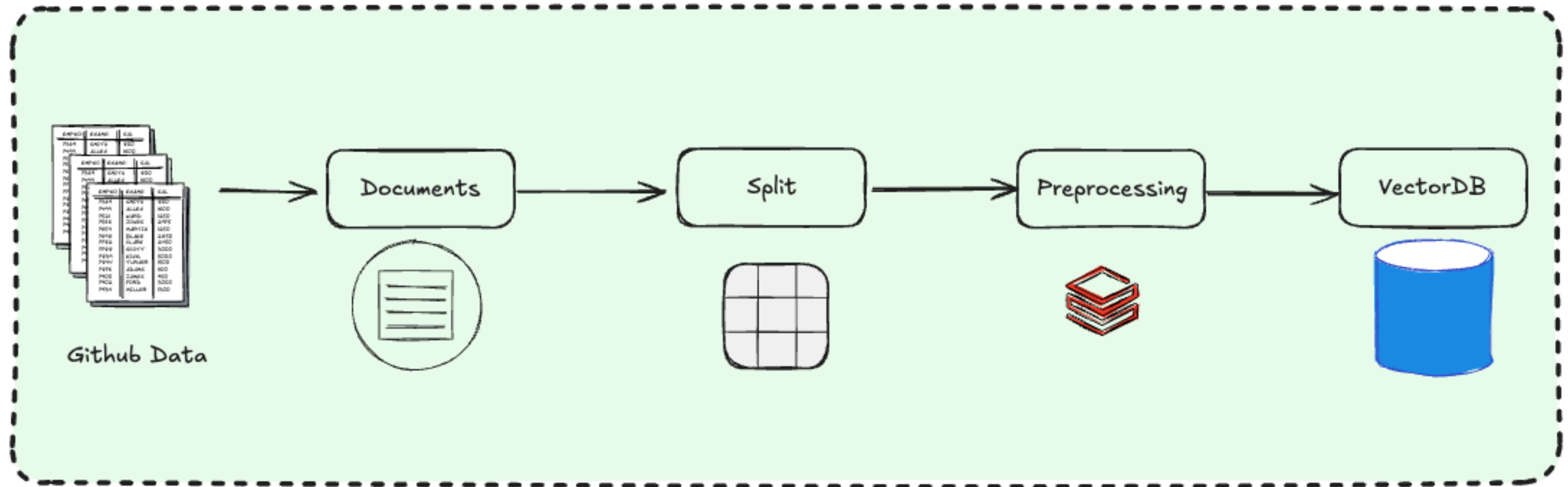
This module takes an user query, and makes a decision what type (Vector DB, Tool) of query the user requested,

The current classification include,

1. Duplocloud Github documents
2. Wikipedia search
3. Brave search



6. Indexing - Chroma Vector DB

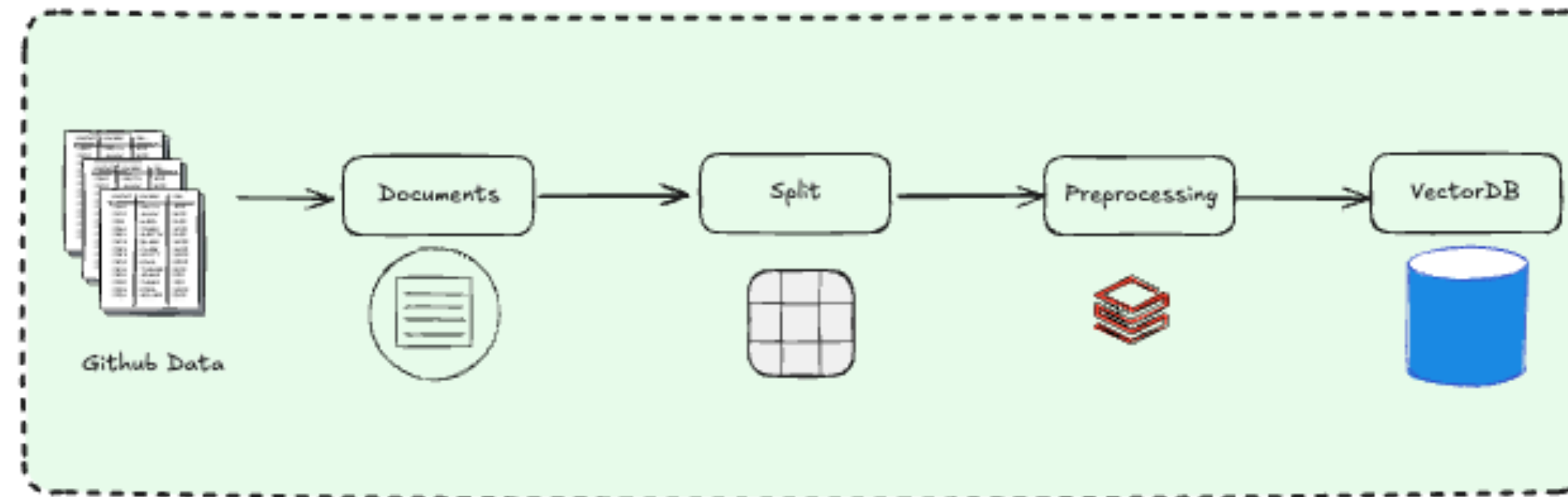


Representing textual or other forms of data as vectors (numerical representations) and then indexing them into a vector database Chroma

This module takes an user query, and makes a decision what type (Vector DB, Tool) of query the user requested, The current classification include,

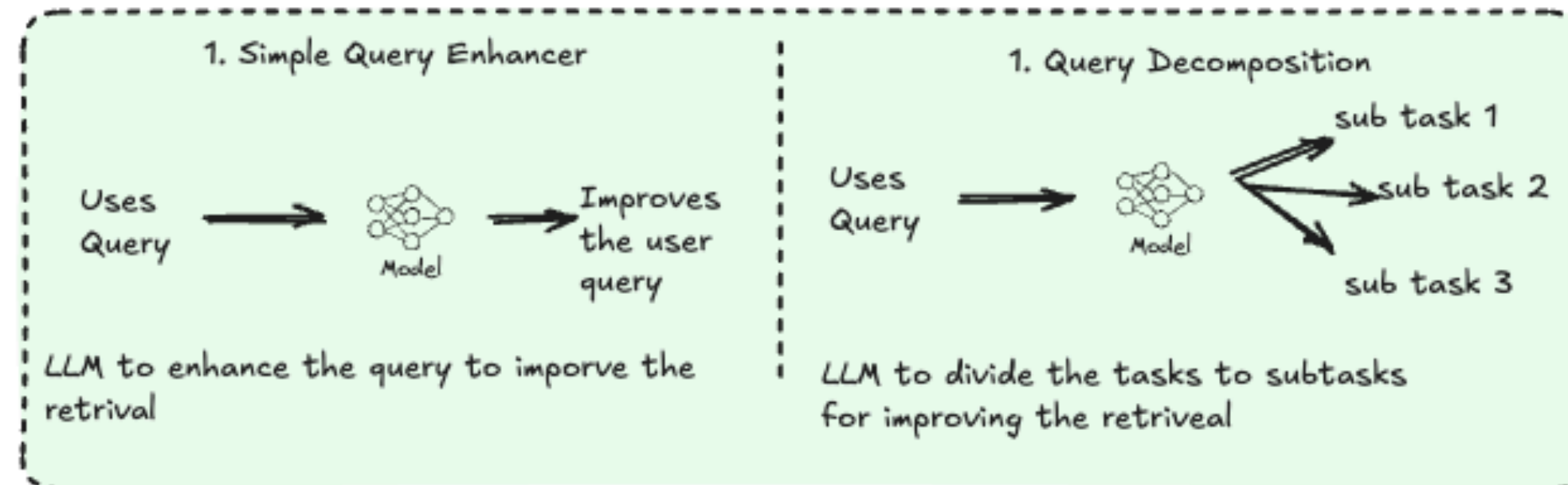
1. Duplocloud Github documents
2. Wikipedia search
3. Brave search

Indexing

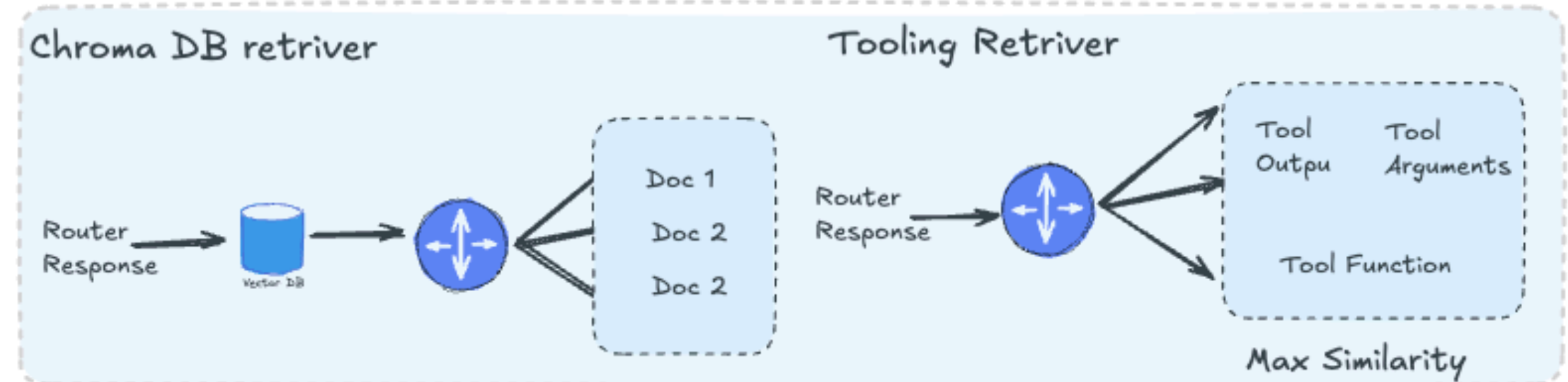


Overall Workflow

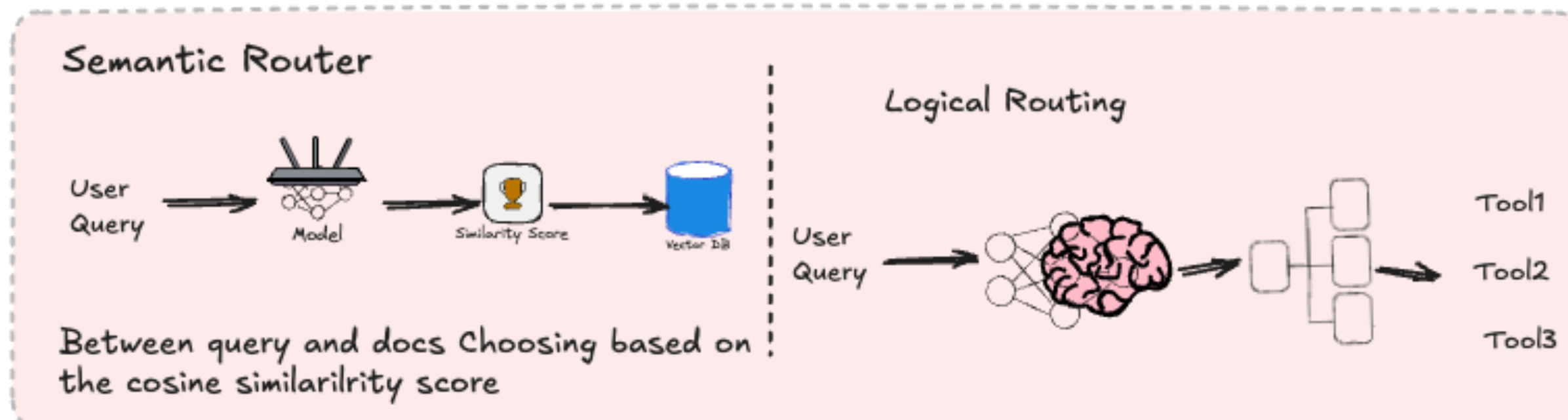
Translator



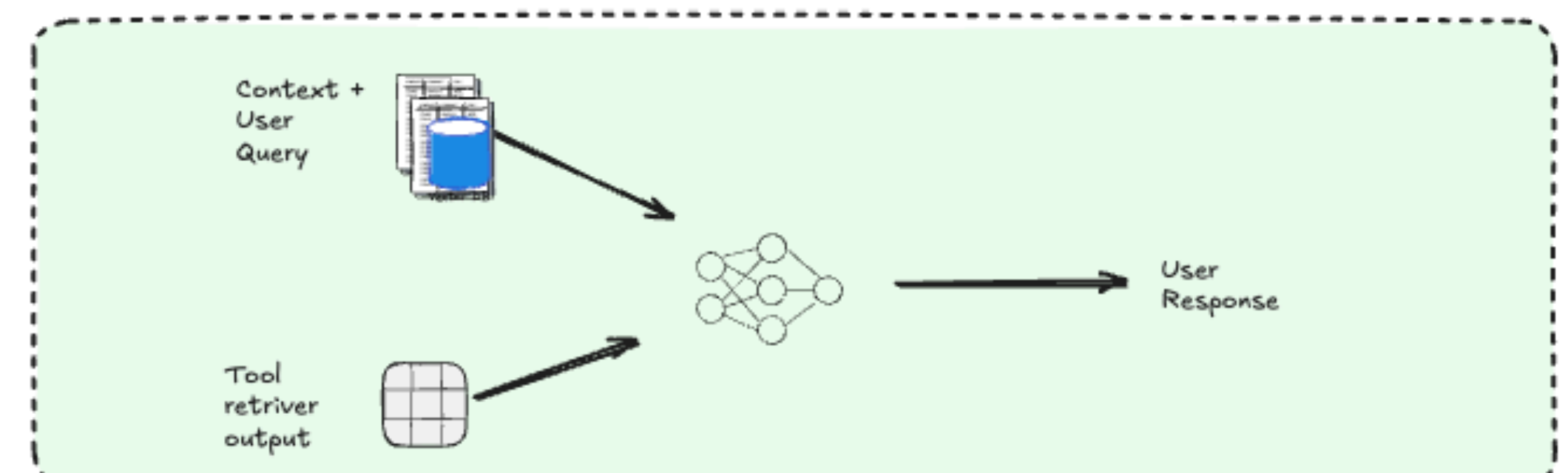
Retriever



Router



Generator



Test Cases

Duplocloud queries

Non-Duplocloud queries

Web-search queries

Real time queries

<https://drive.google.com/file/d/1fh7CKp27c-NoKm7fKq2lhJjRNf3lZMRg/view>

Future Work

1. **Vector DB** - explore different types of vectoredb
2. **Data**
 1. Index different types of data and structure
 2. Data Cleaning
 3. Improving chunking
3. **RAG Pipeline**
 1. **Query Translator**
 1. Implementing various different methods in the query abstraction and query subtasks
 2. Implementing a entire pipeline to perform subtasks or subqueries
 3. Testing the performance of baseline user query vs enhanced queries
 4. Improving query decomposer

Future Work

1. RAG Pipeline

1. Query Router

1. Scaling the router to accommodate **various router paths**
2. Performing testing for real time user queries
3. Improving the **similarity search** for the vectordb data
4. Router mechanism for **multiple vector data bases**
5. Including multiple tools to scale the router mechanism
6. implement Multit-query of the user that can come from query decomposition

2. Query Retrainer

1. Trade-off analysis on the docs retrieved
2. Improving the context
3. Building relevancy module with the query and the docs
4. Rerouting if the documents are not relevant
5. Reranking, Rank fusion mechanism

Future Work

1. RAG Pipeline

1. Query Generator

1. Improving the generation mechanism by testing accuracy - Human testing
2. Using generation quality to query reconstruction or query improvement

2. Data Indexer

1. Including **chunk optimization** techniques for big data
2. **Multi-representation indexing** - Summarizing the chunks including in for improving the retrieval
3. For graph data including tree based or hierarchical indexing techniques

Framework

Python - AI RAG module building

VectorDB - Chroma

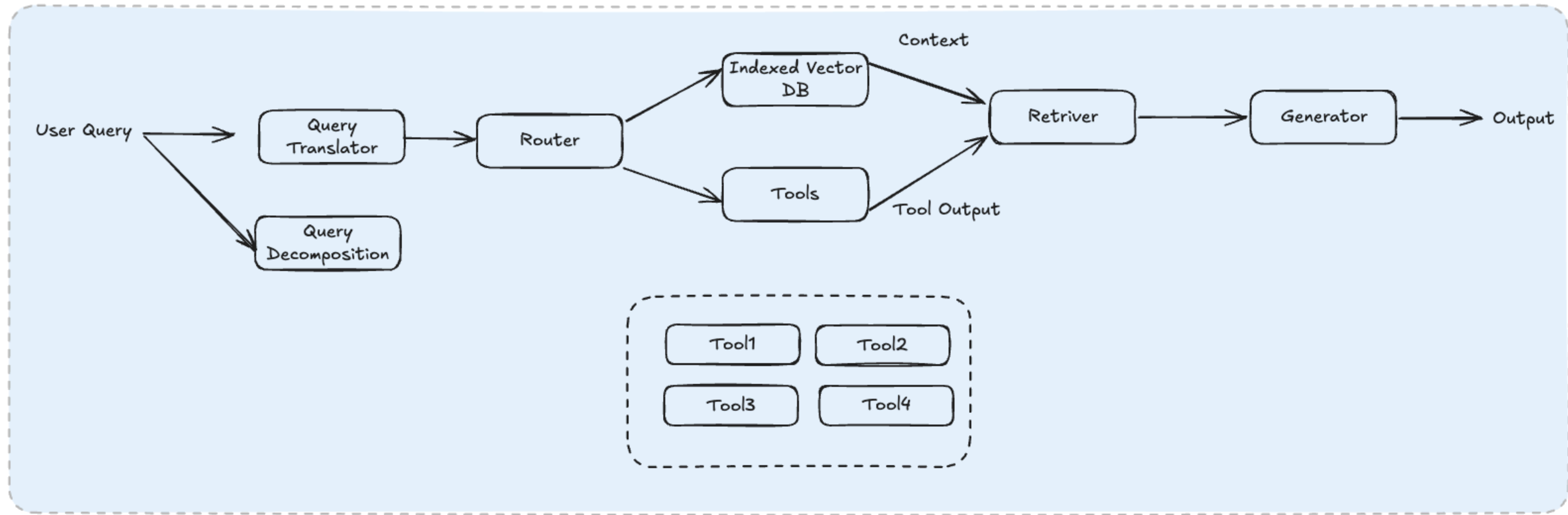
Langchain - Orchestration

RESTful APIs - Fastapi



Appendix

Overall workflow



Data Indexing Workflow

