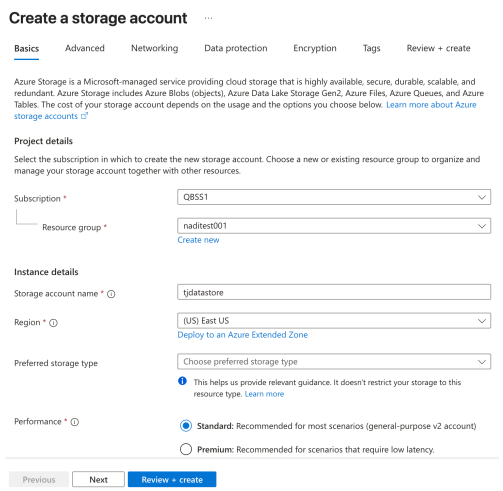
Prereqs for RAG with Azure AI Search

Azure

i. Create a Storage account — *acasdatastore*
ii. Create Embedding Deployment — embeddingacasrag
iii. Create Chats Deployment — chatmodelrag
iv. Create Azure AI Search Service — azureaisearchrag
v. Using the chats Deployment
    i. Upload the documents.
vi. Ingestion > preprocessing > Indexing

RAG PREREQUISITES

1. Storage Account



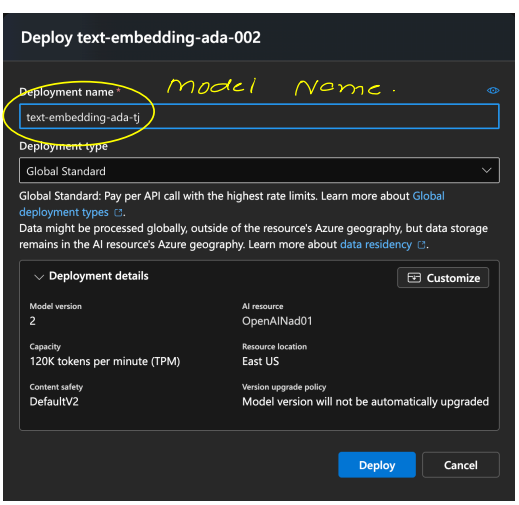
Redundancy

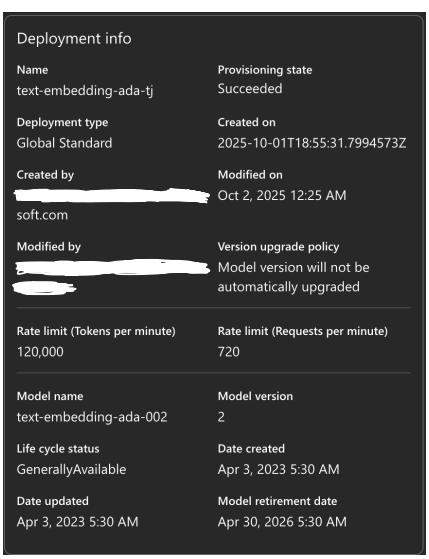
↳ Stores copies of Data.

2. Deploy Embedding Model

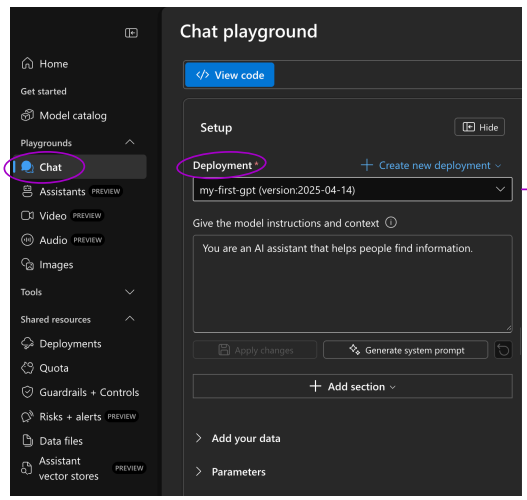
Deployments → Model

Model = 3 (large / small / Ada.)





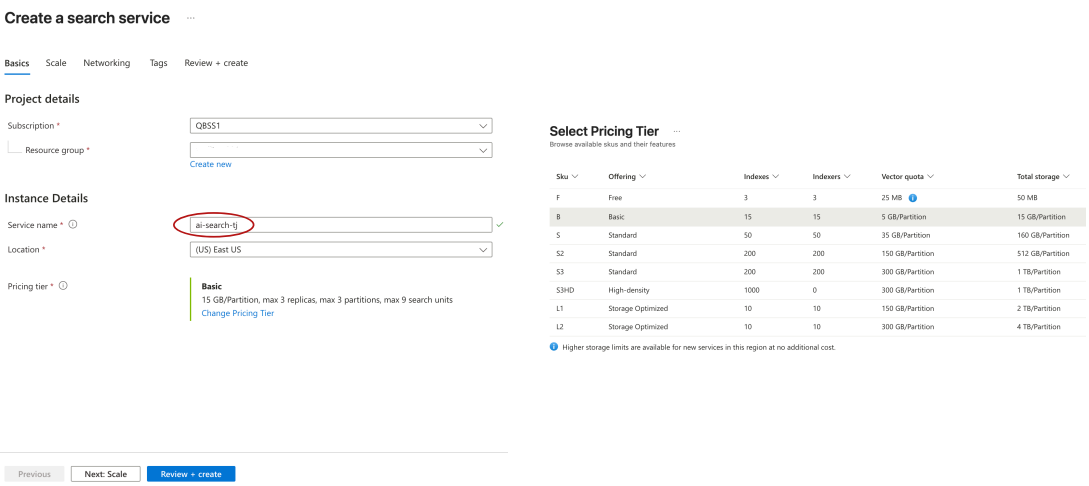
3. Chat Completion Deployment



→ Chat Deployment

4. Create AI Search

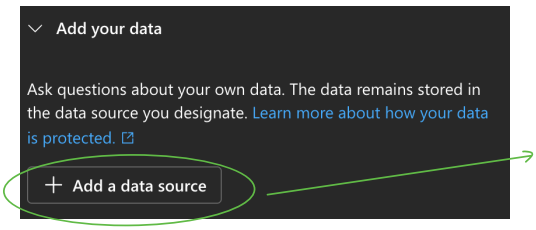
Azure portal → AI Search

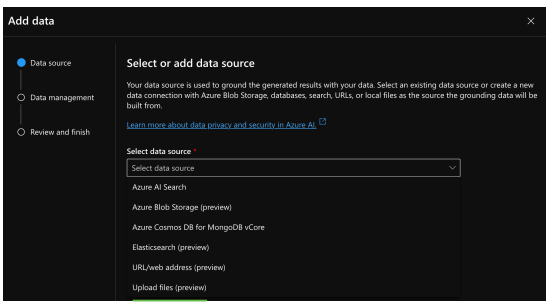


So now we have ;

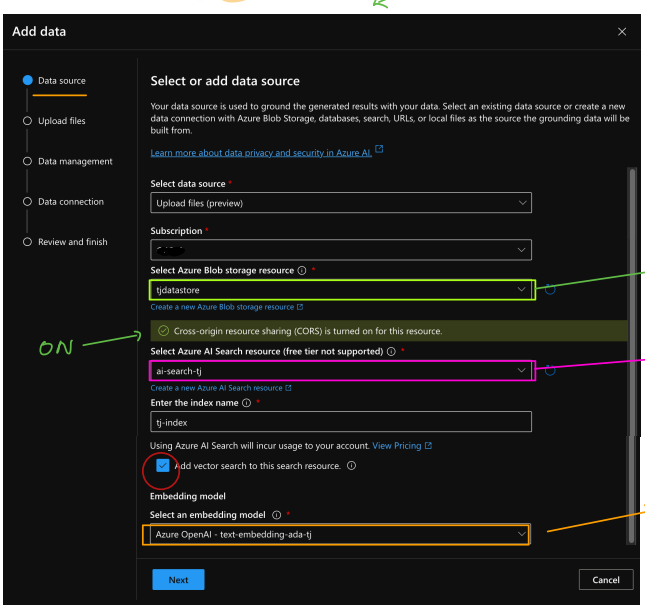
- Storage Account
- Embedding Deployment
- Azure AI Search
- Chat Model Deployment

5. ADD DATA





Data Sources



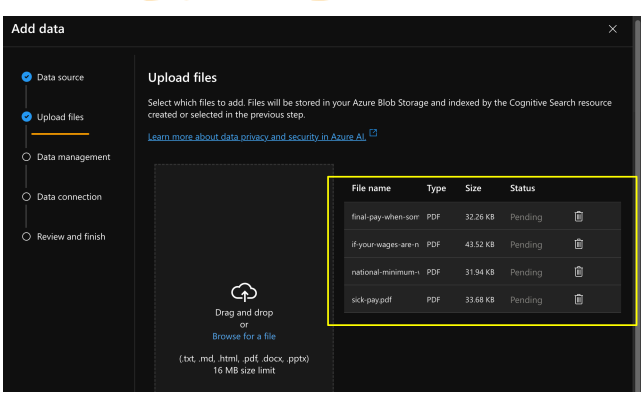
ON

Storage Account

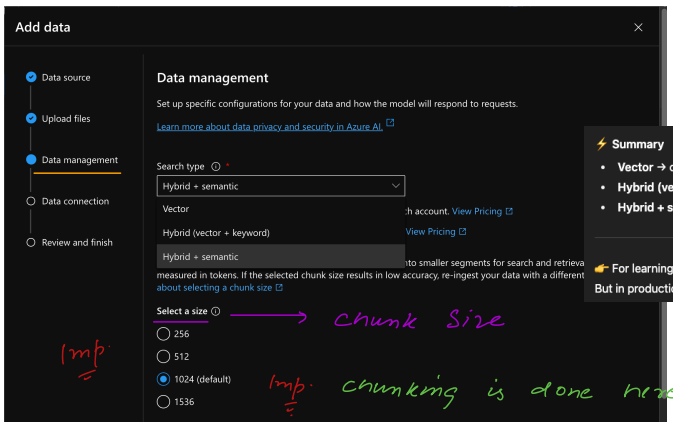
AI Search

Embedding Deployment

Upload Files



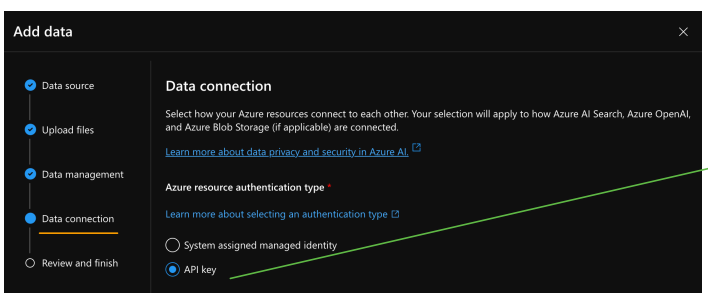
Data Management



Summary

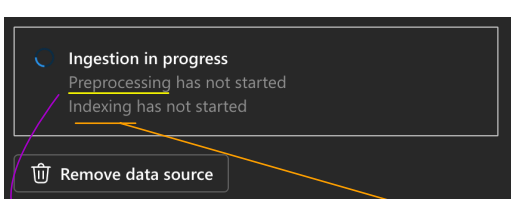
- **Vector** → only embeddings, best for semantic similarity.
- **Hybrid (vector + keyword)** → mix of keyword precision + vector meaning.
- **Hybrid + semantic** → best quality, adds AI semantic re-ranking for natural answers.

👉 For learning, you can start with **Vector** to understand embeddings.
But in production, **Hybrid + semantic** usually gives the most natural, Google-like search experience.



Data Ingestion

once Review & finish ; Data Ingestion starts



preprocessing Then Indexing

1. Preprocessing (data preparation)

- Azure runs **document cracking**:
  - Extracts text from PDFs, Word, PPT, HTML, scanned images (OCR).
  - Normalizes into a structured format.
- Cleans and detects fields (title, content, author, metadata).
- Optionally runs **cognitive skills** if you add them:
  - Language detection
  - Entity recognition
  - Key phrase extraction
  - Image tagging, OCR for scanned docs

👉 End result = clean, structured chunks of content + metadata.

2. Indexing (make searchable)

- Content is **pushed into the Search Index**.
- For each document/chunk, Azure creates:
  - **Full-text index** (keyword/BM25 search)
  - **Vector embeddings index** (if you've enabled vector search → usually via Azure OpenAI embedding model)
  - Metadata fields (filters, categories, tags)

👉 End result = a searchable index that can support keyword search, vector similarity, or hybrid search.

Q. what indexing method is used in Azure AI Index?

HNSW → Hierarchical Navigable Small world Graph. (Graph based)

→ It uses ANN (Approx. nearest neighbours).

✅ Key Difference in Indexing Context

Feature	KNN	ANN
Type of search	Exact	Approximate
Speed	Slow for large datasets	Fast
Accuracy	100%	Usually <100% (tunable)
Memory & CPU	High for large datasets	Optimized with index structures
Index structures	Usually none (flat)	HNSW, IVF, PQ, etc.

Imp. Q. what is Hybrid Search?

Hybrid = Both Keyword + Vector search
BM25 HNSW

Hybrid search combines:

1. **Vector Search (Semantic / Embedding-based)**
   - **Goal:** Find semantically similar items.
   - **Index used:** ANN (like **HNSW**, IVF, or PQ).
   - **Input:** Query embedding vs document embeddings.
   - **Output:** Candidate vectors that are "close" in semantic space.
2. **Keyword Search (Exact / Lexical)**
   - **Goal:** Match query terms exactly or partially.
   - **Method:** **BM25** (based on bag-of-words).
   - **Input:** Query terms vs document terms.
   - **Output:** Ranked documents based on term frequency, IDF, and document length.

How Hybrid Works

1. Query comes in.
2. Two parallel searches:
   - **ANN search** for embeddings → returns semantically similar items.
   - **BM25 search** for keywords → returns lexically matching items.
3. **Merge / rerank:** Combine the results from both methods using a scoring function.
   - Example: weighted sum of BM25 score + ANN similarity score.

Key Takeaways

- **ANN (HNSW)** → finds semantic neighbors, **fast but approximate**.
- **BM25 (BOW)** → finds lexical matches, **exact but shallow**.
- **Hybrid = best of both worlds** → captures both meaning and keywords.

Working of BM25 and HNSW

BM25

- **Type:** Keyword-based search (bag-of-words).
- **How it works:** Scores documents based on **term frequency**, **inverse document frequency**, and **document length**.
- **Example:**
  - Query: `"data cleaning"`
  - Document A: `"Data cleaning techniques for Python"` → higher score
  - Document B: `"Machine learning in Python"` → lower score

TF-IDF → more frequent words, unique words → Rank high

HNSW (ANN) Graph – Semantic Aware

- **Type:** Approximate nearest neighbor search for **vectors**.
- **How it works:** Builds a **graph of vectors**, traverses efficiently to find nearest neighbors in high-dimensional space.
- **Example:**
  - Query embedding: vector of `"data cleaning"`
  - Returns documents semantically similar even if keywords differ, e.g., `"Python preprocessing methods"`

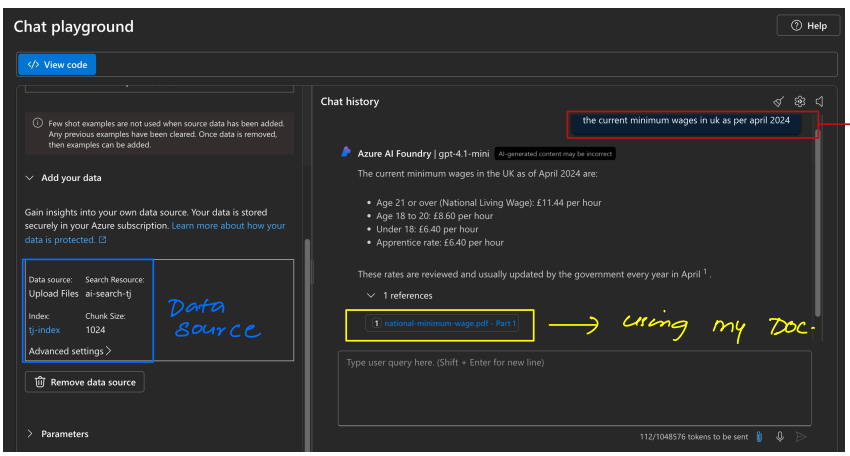
Key Difference

- BM25 → exact keyword match.
- HNSW → semantic similarity using embeddings.

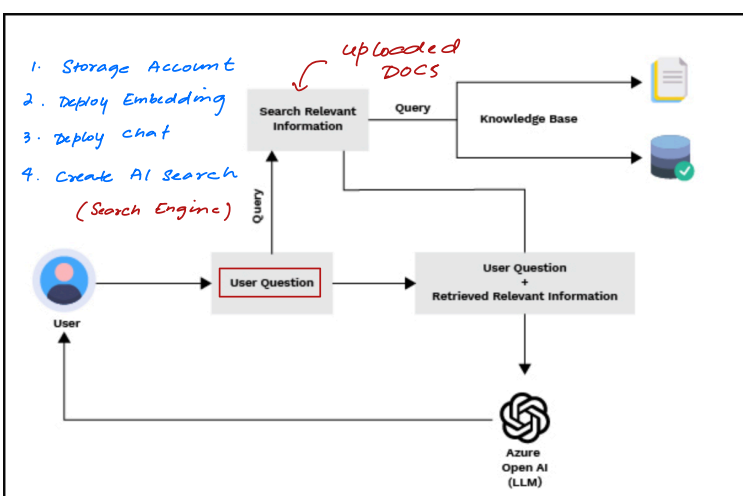
Q1 Why indexing ?

For fast and efficient retrieval of Data.
Filtering etc.

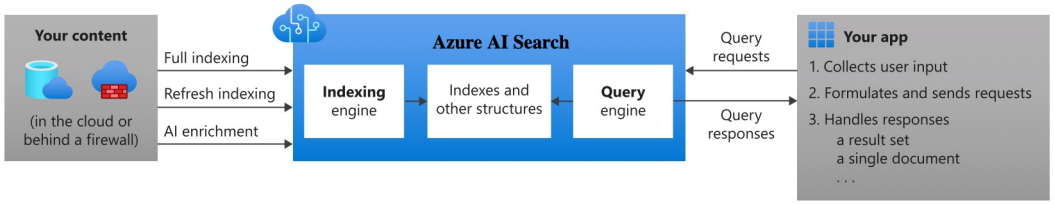
CHAT PLAYGROUND WITH OWN DATA

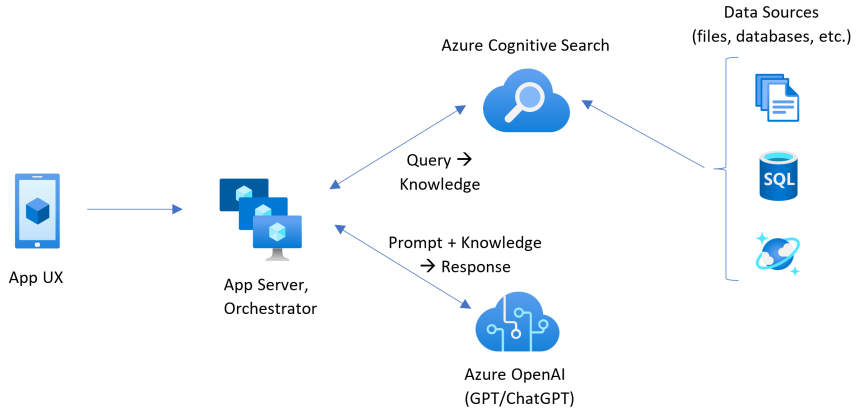


Complete process



Azure AI Search





Azure AI Search (formerly known as "Azure Cognitive Search") provides secure information retrieval at scale over userowned content in traditional and generative AI search applications.

Data Chunking

Instead of treating the entire document as a single unit, chunking divides it into smaller segments, typically based on paragraphs, sections, or sentences.

AI Powered Indexing:

Utilizes AI to automatically extract and enrich data from a variety of sources including documents, images, and media files.

Semantic Search:

Uses advanced machine learning models to understand the **intent** behind queries, providing more relevant results.

Cognitive Skills:

Integrates predefined cognitive skills such as optical character recognition (**OCR**), language **detection**, and entity recognition or allows the creation of custom skills.

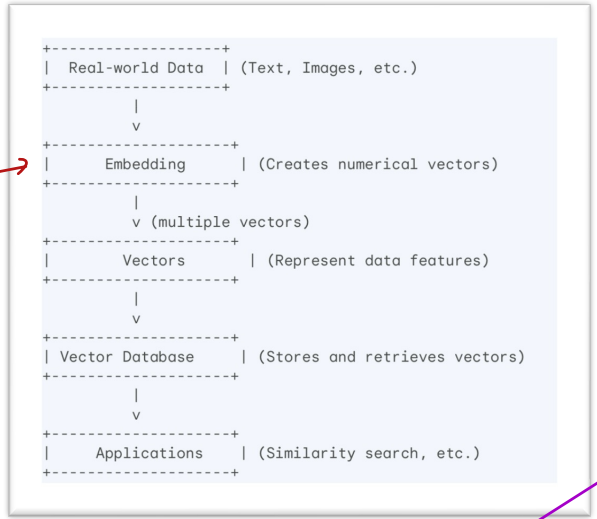
Scalable Infrastructure:

Automatically scales to accommodate data volume and query traffic, ensuring efficient performance without manual intervention.

Integration with Azure Ecosystem:

Works seamlessly with other Azure services, such as Azure AI, Azure Functions, and more, for comprehensive data processing and handling.

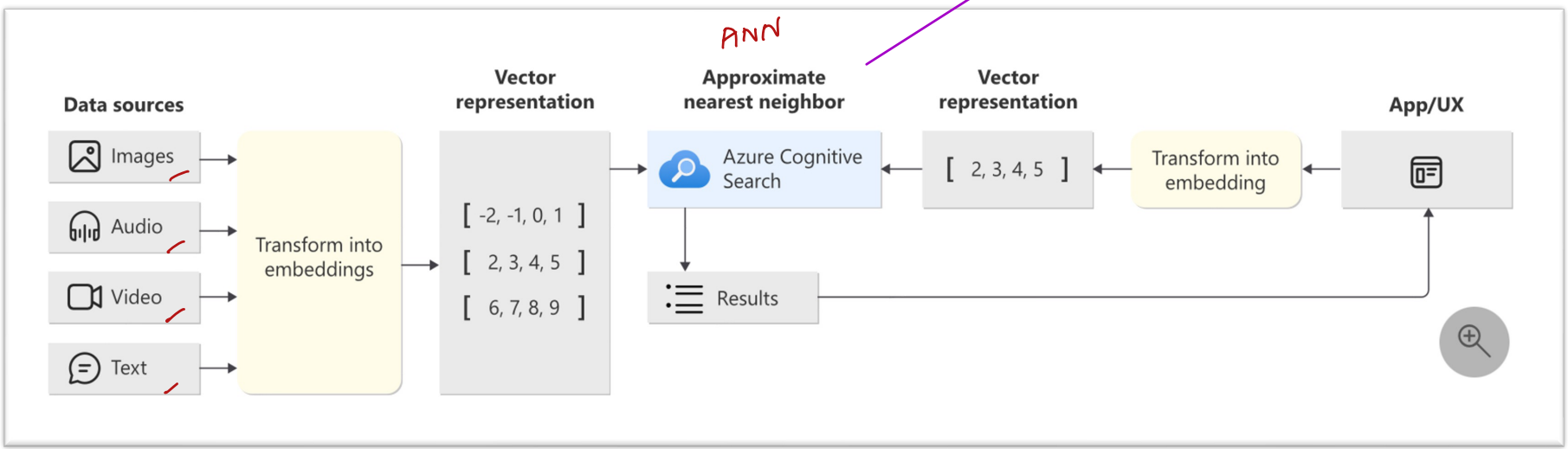
How vector search works in Azure AI Search

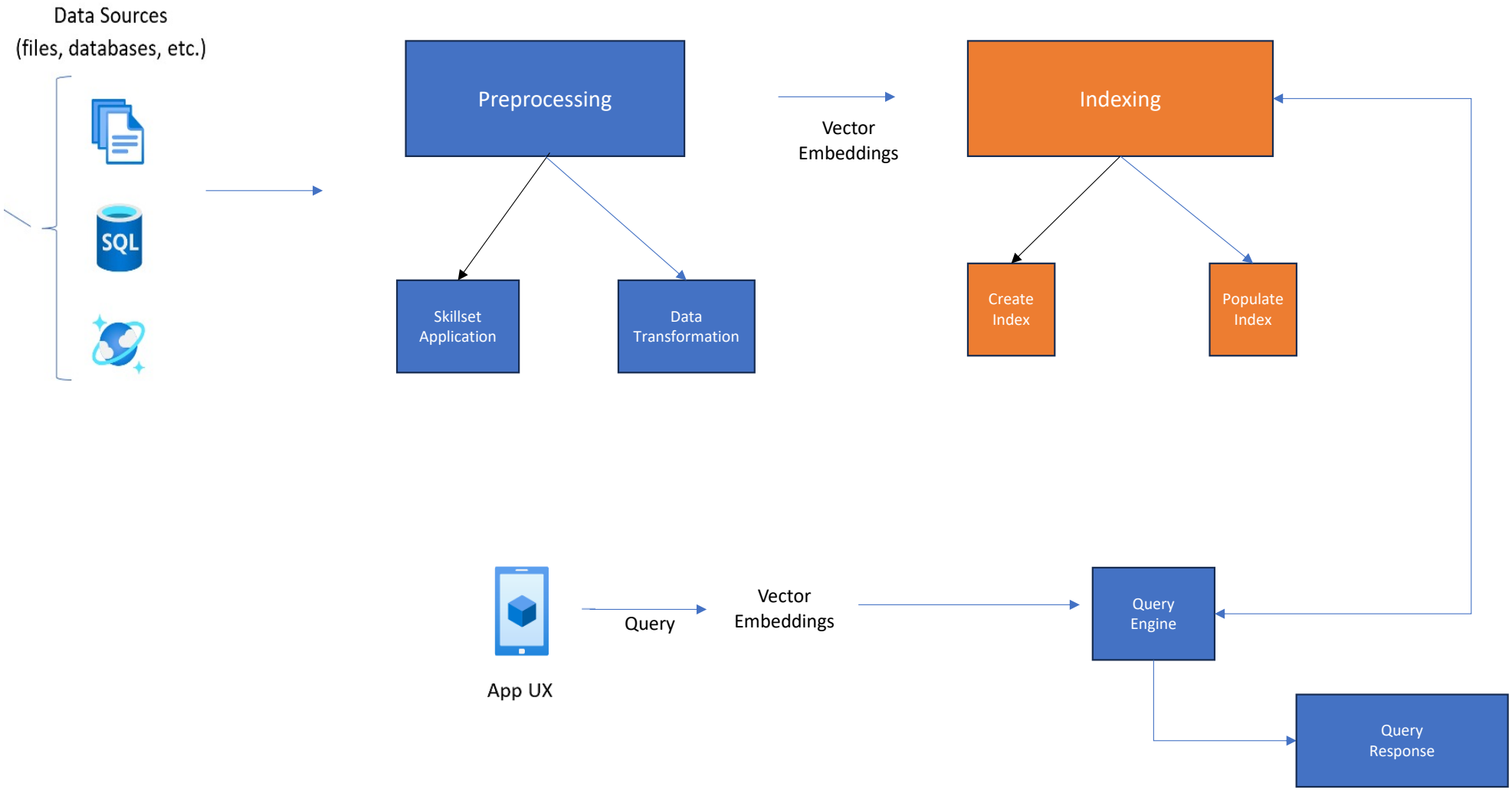


ADA

Relevance Ranking

ANN




Data Sources
(files, databases, etc.)
SQL
Preprocessing
Skillset Application
Data Transformation
Vector Embeddings
Indexing
Create Index
Populate Index
App UX
Query
Vector Embeddings
Query Engine
Query Response