PART C – Search & Retrieval (RAG) System – Documentation

1. Problem Statement

The goal is to implement a semantic search system that retrieves the most relevant historical emails based on a user query.

This forms the backbone of Retrieval-Augmented Generation (RAG) workflows inside customer support intelligence systems.

---

2. Approach Overview

We built a dense embedding–based search pipeline using:

- Sentence-Transformers for embeddings
- FAISS for similarity search
- Custom reasoning + confidence scoring

All operations run completely **locally**, with no external API usage.

---

3. Model & Embedding Strategy

Embedding Model:

all-MiniLM-L6-v2

Chosen for:

- Excellent performance vs size
- Fast CPU inference
- High semantic accuracy
- Suitable for large-scale retrieval

Embedding Properties:

- 384-dimensional
- L2-normalized
- Supports cosine similarity (via inner product)

---

4. Indexing with FAISS

Why FAISS?

- High-speed nearest-neighbor search
- GPU-ready (CPU used here)
- Optimized for vector similarity lookup

Index used:

IndexFlatIP (Inner Product)

Because cosine similarity = inner product when vectors are normalized.

---

5. Retrieval Pipeline

Step-by-step:

1. **Encode all emails** → embedding matrix
2. **Build FAISS index**
3. **Encode query**
4. **Retrieve top-k similar emails**
5. **Return structured result:**
    - top match
    - tag
    - similarity score
    - confidence
    - reasoning
    - alternate matches

---

6. Example Output

Query: “unable to assign emails automatically”

Top Result:

- Email: Auto-assign slow…
- Score: 0.6906
- Tag: automation_delay

Confidence: 0.846

Reasoning: "The top matched email contains similar context..."

Alternates: [{email, tag, score}, ...]

This demonstrates accurate retrieval even with short inputs.

---

7. Error Analysis

1. Short noisy emails

Some inputs (e.g., “please help”) match many unrelated emails.

2. Domain mismatch

The model is not fine-tuned on customer support tickets.

3. No ranking re-weighting

All matches rely solely on cosine similarity.

4. No customer isolation

Like Part A, future versions should limit retrieval to customer-specific history.

---

8. Production Improvement Ideas

1. Fine-tune embeddings on Hiver support data

Massively improves relevance and domain understanding.

2. Add hybrid retrieval

Merge:

- keyword search
- dense vector search
- metadata filtering (customer, category)

3. Add LLM summarization layer

After retrieving results, an LLM can:

- Summarize relevant emails
- Provide recommended actions
- Explain root cause