BFSI AI Assistant — Technical Documentation

1. Introduction

The BFSI AI Assistant is a multi-stage AI system built to handle customer queries in the banking and financial services domain. The primary design goal was to create a solution that balances **accuracy, speed, and reliability** while running fully on local infrastructure.

Instead of relying on a single model for all tasks, my system uses a **tiered architecture** where each stage is optimized for a specific type of query. This approach reduces hallucinations, improves response time, and ensures that policy-related answers are grounded in real documents.

2. Design Logic

The system is based on three key principles:

1. Use retrieval whenever possible to ensure correctness
2. Use generation only when reasoning is required
3. Use knowledge grounding for policy or regulatory queries

3. Models Used and Rationale

3.1 Embedding Model — all-MiniLM-L6-v2

Where it is used

- FAQ similarity matching (Tier-1)
- Document retrieval in RAG (Tier-3)

Why this model

- Lightweight (fast on CPU)
- Produces high-quality semantic embeddings
- Low memory footprint (ideal for local setup)

Role in pipeline

Converts text queries and documents into vector representations so semantic similarity can be computed.

3.2 Small Language Model — TinyLlama-1.1B (GGUF Quantized)

Where it is used

- Tier-2 reasoning
- Tier-3 answer generation

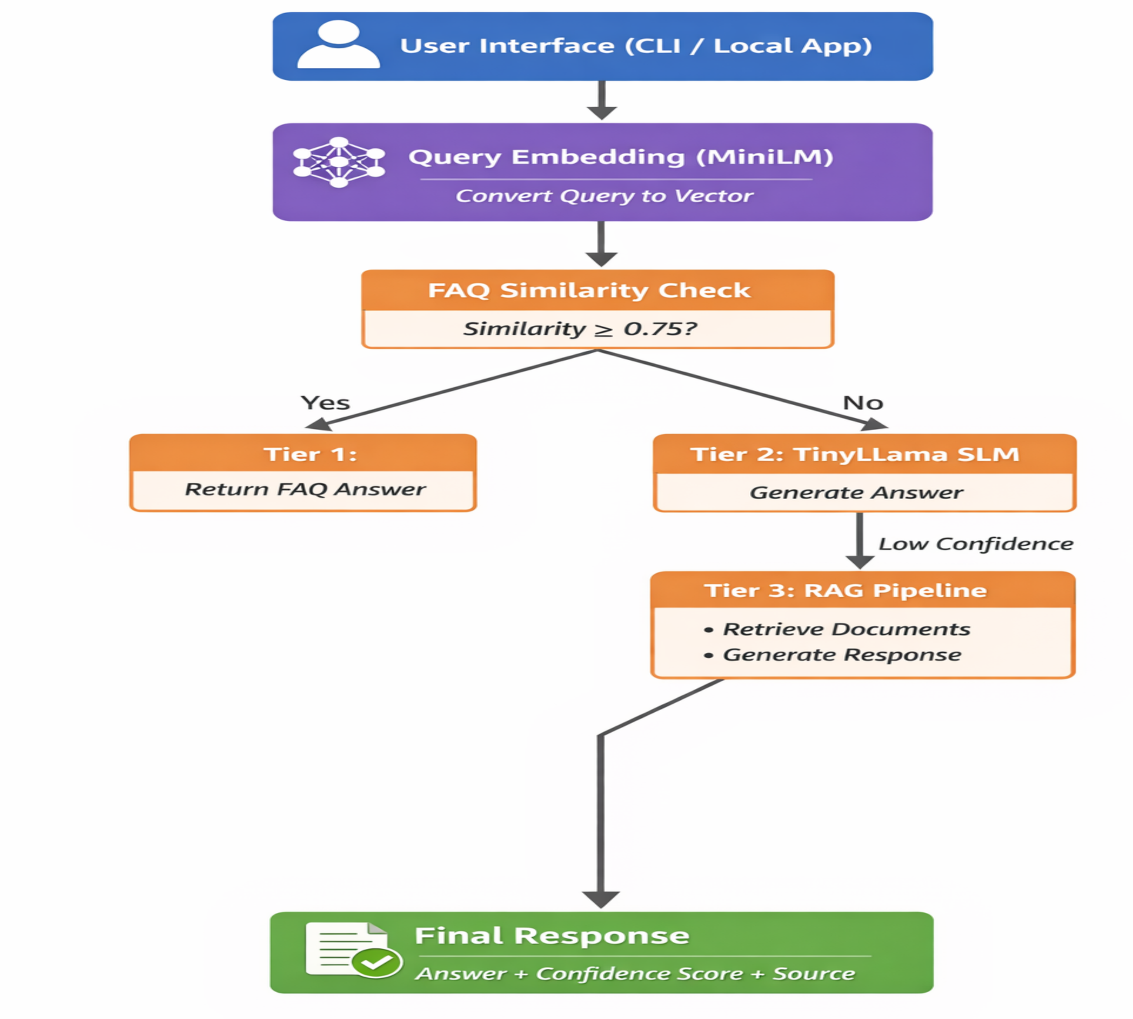
Why this model

- Small enough to run locally (≈600 MB quantized)
- Instruction-tuned for conversational responses
- Low latency compared to larger LLMs

Role in pipeline

Generates natural language responses when retrieval alone is insufficient.

4. Architecture Overview



5. Tiered Processing Logic

5.1 Tier 1 — FAQ Semantic Retrieval

Purpose

Provide instant answers for frequently asked questions.

Process

1. Query converted to embedding
2. Cosine similarity computed with FAQ dataset
3. Highest scoring match selected

Threshold

```
SIM_THRESHOLD = 0.75
```

Why 0.75

Empirically balances precision and recall.
Below this value, semantic matches may be too weak to trust.

Output

Direct deterministic answer from dataset.

5.2 Tier 2 — Local Reasoning (SLM)

Purpose

Handle queries that require procedural explanation but do not need policy grounding.

Activation Condition

```
SLM_THRESHOLD = 0.45
```

This means:

- Query is somewhat related to known domain
- But not close enough for FAQ retrieval

Why 0.45

Ensures the model is invoked only when semantic confidence exists, preventing irrelevant generation.

Prompt Design

The prompt includes:

- System instructions
- User query
- Response constraints

This helps control verbosity and avoid hallucinations.

5.3 Tier 3 — Retrieval Augmented Generation

Purpose

Answer policy, regulatory, and domain-specific queries using knowledge grounding.

Process

1. Query embedding generated
2. FAISS retrieves top-k documents(k=3)
3. Retrieved context injected into prompt
4. TinyLlama generates a grounded answer

Why top_k = 3

Provides enough context without overwhelming the model or increasing latency.

6. Vector Database Configuration

Engine: FAISS

Index Type: Flat L2

Reason

- Fast similarity search
- Suitable for small-to-medium datasets
- No training required

7. Routing Algorithm

The routing decision follows a cascading logic:

1. If FAQ similarity ≥ 0.75 → Tier-1
2. Else if similarity ≥ 0.45 → Tier-2

3. Else → Tier-3

This ensures:

- Fastest path for simple queries
- Controlled generation for complex queries
- Knowledge grounding for policy queries

8. Parameters Summary

Parameter	Value	Purpose
FAQ threshold	0.75	Ensure strong semantic match
SLM threshold	0.45	Trigger reasoning layer
Top-k retrieval	3	Context size for RAG
Embedding dimension	384	Vector representation size
Context window	2048 tokens	LLM input limit

9. Performance Profile

The system demonstrates:

- Sub-second responses for FAQ queries
- Moderate latency for LLM responses
- Low memory usage due to quantized model

This makes it suitable for edge or on-prem deployment.

10. Implementation Structure

```
app/
├── pipeline.py      → routing logic
├── similarity.py    → semantic search
├── slm.py           → TinyLlama inference
└── rag/
    ├── ingest.py
    ├── retrieve.py
    └── generate.py
```

Each module is isolated to maintain separation of concerns.

11. Why This Architecture Works Well

- Reduces unnecessary LLM calls
- Improves factual correctness
- Maintains low compute cost

- Provides explainable routing decisions

This is aligned with best practices in enterprise GenAI systems.

12. Conclusion

The BFSI AI Assistant demonstrates a practical and scalable approach to building domain-specific AI systems. By combining semantic retrieval, lightweight local language models, and retrieval-augmented generation, the system ensures both efficiency and reliability — two critical requirements in financial applications.