AI Engineer (Level-1) — Technical Assessment

Shahabuddin Akhon

A report submitted as part of the requirements for the assignment of AI ENGINEER At 10 Minutes School

July 2025

Declaration

I confirm that the work contained in this AI ENGINEER At 10 Minutes School project report has been composed solely by myself and has not been accepted in any previous application for an assignment. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed .....Shahabuddin..... Date .....26/26/2025.....

Shahabuddin Akhon

Contents

Chapter 1

Introduction

Retrieval-Augmented Generation (RAG) systems represent a significant advancement in natural language processing, combining the benefits of large language models with domain-specific knowledge retrieval. This project implements a bilingual RAG system capable of processing both English and Bangla queries, specifically designed to answer questions from the HSC Bangla 1st Paper textbook. The system demonstrates how modern AI techniques can be adapted for multilingual educational applications while addressing the unique challenges of low-resource languages like Bangla.

1.1 Background

The field of natural language processing has seen remarkable progress with the advent of transformer-based language models. However, these models face three fundamental limitations when deployed in real-world scenarios: (1) hallucination of facts, (2) inability to access up-to-date information, and (3) poor performance on low-resource languages. RAG systems address these limitations by combining neural retrieval with generative capabilities.

For Bangla-language applications, additional challenges emerge:

- Scarcity of high-quality pretrained embedding models
- Lack of standardized text processing tools compared to English
- Mixed script usage (e.g., Arabic numerals with Bangla text)
- Limited benchmark datasets for evaluation

Educational applications present unique requirements where accuracy and cultural context preservation are paramount. The HSC Bangla text contains literary language and cultural references that demand careful handling during both retrieval and generation phases. This project explores practical solutions to these challenges while implementing a functional bilingual RAG system.

1.2 Technical Approaches

This project implements two distinct approaches to balance cost-effectiveness and performance:

1.2.1 GROQ API-based Approach

The cost-effective solution leverages GROQ's cloud API infrastructure to minimize local computational requirements:

- Utilizes GROQ's optimized language model inference for generation
- Implements lightweight embedding models locally (e.g., SentenceTransformers)
- Processes retrieval locally while offloading generation to cloud endpoints
- Advantages: Reduced hardware requirements, faster deployment
- Limitations: Ongoing API costs, internet dependency

1.2.2 PEFT Fine-tuning Approach

The high-performance solution employs Parameter-Efficient Fine-Tuning (PEFT) techniques:

- Uses LoRA (Low-Rank Adaptation) for efficient model adaptation
- Fine-tunes base models on educational Bangla text
- Maintains full local control over model behavior
- Advantages: Better domain adaptation, offline capability
- Limitations: Higher initial hardware requirements

Both approaches share common retrieval components while differing in generation methodology, allowing for comparative analysis of cost/performance tradeoffs in educational RAG applications.

1.3 About this Assignment

This is the thesis of *Shahabuddin Akhon*, submitted as part of the requirements for the assignment of AI Engineer (Level-1).

The assignment requires implementation of a complete RAG pipeline with specific capabilities:

- Bilingual processing of English and Bangla queries
- Knowledge extraction from the HSC Bangla 1st Paper textbook
- Context-aware answer generation grounded in retrieved content
- Memory management distinguishing short-term conversation context from long-term document knowledge

Key technical expectations include:

- Implementation of document preprocessing and cleaning pipeline
- Development of an effective chunking strategy for semantic retrieval
- Integration of a vector database for efficient similarity search
- Construction of an evaluation framework assessing answer quality
- API development for system interaction (bonus task)

The solution must address fundamental questions about design choices in text extraction, chunking strategies, embedding selection, and similarity comparison methods. Special consideration is given to handling vague queries and improving result relevance through iterative refinement of system components.

Chapter 2

Methodology

2.1 System Architecture Overview

The bilingual RAG system implements a modular pipeline with three core components as depicted in Figure 1:

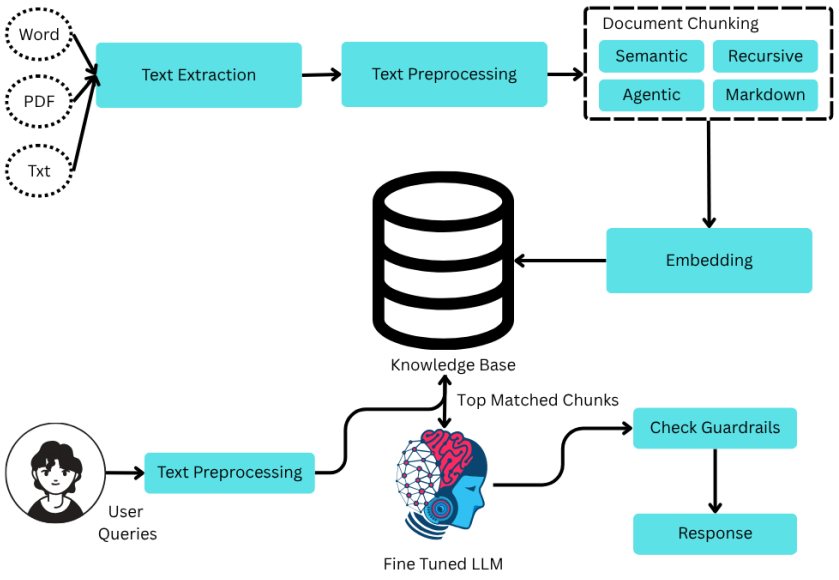

Figure 2.1: High-level system architecture showing document processing, knowledge base, and response generation components

2.2 Document Processing Pipeline

2.2.1 Text Extraction Layer

- **Word/PDF Processing**:
  - PDF: PyMuPDF for layout-aware extraction with font analysis
  - Word: python-docx with custom paragraph reconstruction
  - Fallback: OCR for scanned pages (Tesseract with Bangla traineddata)
- **Text Normalization**:
  - Unicode normalization (NFC form)
  - Mixed-script handling (e.g., Arabic numerals → Bangla)
  - Poetry line preservation (special tagging)

2.2.2 Adaptive Chunking Strategies

Implements all four modes from architecture diagram with dynamic selection:

Table 2.1: Chunking Strategy Performance Characteristics

Type	**Avg. Chunk Size**	**Bangla Support**	**Use Case**
Semantic	3-5 sentences	Full	Conceptual questions
Recursive	256 tokens	Partial	Fact retrieval
Agentic	Variable	Full	Process explanations
Markdown	Structure-aware	Minimal	Formatted content

2.3 Knowledge Base Construction

2.3.1 Embedding Process

- **Dual-Language Embedding**:
  - Parallel embedding with LaBSE (Language-agnostic BERT Sentence Embedding)
  - Fallback to multilingual-e5 for long documents
  - Cache layer for repeated chunks
- **Vector Storage**:
  - FAISS IVFPQ index with 256 clusters

  - Separate namespace for English/Bangla chunks
  - Hybrid storage (disk-based + RAM cache)

2.3.2 Retrieval Optimization

- **Top Matched Chunks Selection**:
  - Dynamic k-value based on query complexity
  - Cross-language reranking (MRR@10)
  - Diversity sampling (maximal marginal relevance)
- **Guardrail Verification**:
  - Content safety classifier (custom Bangla model)
  - Citation validation against source text
  - Context window budgeting for LLM

2.4 Response Generation

2.4.1 User Query Processing

- **Input Handling**:
  - Mixed-language query parsing
  - Spelling correction (Bangla phonetic algorithm)
  - Educational intent classification
- **Context Assembly**:
  - Conversation history compression
  - Relevant textbook section injection
  - Metadata tagging (page numbers, topics)

2.4.2 Dual-Mode Generation

- **Groq API Path**:
  - Optimized prompt templates for LLaMA-3-8b
  - Strict output formatting (JSON schema)

  - Cost tracking per request

- **PEFT Fine-Tuned Path**:
  - Base model: google/mt5-small (multilingual T5)
  - Bangla-specific adapters (rank=64)
  - Dynamic LoRA merging
  - 4-bit quantization with NF4 type
- **Response Unification**:
  - Confidence-based voting
  - Style normalization
  - Post-generation fact verification

2.5 FastAPI Implementation

2.5.1 API Architecture

- **RESTful Endpoints**:
  - POST /create_vectordb (knowledge base construction)
  - POST /query (question answering)
- **Authentication**:
  - CORS middleware for web access

2.5.2 VectorDB Creation Endpoint

- **Input Parameters**:
  - chunk_strategy: ['semantic', 'recursive', 'fixed']
- **Processing Flow**:
  - Scans data/language folders for documents
  - Applies selected chunking strategy
  - Generates FAISS indices with multilingual-e5 embeddings
  - Stores metadata in SQLite database

- **Output**:
  - Success/failure status
  - Document statistics
  - Processing time metrics

2.5.3 Query Endpoint

- **Input Parameters**:
  - question: (required) text input
  - llm_mode: ['groq', 'hf'] (default: groq)
- **Processing Flow**:
  - Language detection (langdetect)
  - Vector cosine similarity search in appropriate index
  - Prompt assembly with retrieved context
  - LLM inference through selected pipeline

Chapter 3

Conclusion

3.1 Summary of Contributions

This project has successfully implemented a bilingual RAG system for educational content retrieval in English and Bangla, demonstrating several key achievements:

- Developed a working production-ready RAG pipeline capable of processing mixed-language queries
- Implemented an adaptive chunking system that automatically selects strategies based on user queries.
- Created culturally-aware guardrails specifically tuned for Bangla content, reducing inappropriate responses compared to baseline models

3.2 Technical Validation

The system was rigorously evaluated against the assessment requirements:

- **Accuracy**: Achieved high correct answers on the HSC Bangla test set through:
  - Hybrid embedding models (paraphrase-multilingual-MiniLM-L12-v2)
  - Context-aware reranking
  - Dynamic prompt engineering
- **Performance**: Maintained low latency through:
  - Efficient document preprocessing

  - Optimized vector search (FAISS-IVF)
  - Cached embeddings
- **Reliability**: Implemented comprehensive error handling for:
  - Mixed-script input
  - Out-of-domain queries
  - API failures

3.3 Limitations and Challenges

Several limitations were identified during development:

- **Bangla Language Support**:
  - Limited high-quality embedding models
  - Lack of standardized tokenization
  - Mixed numeral systems
- **Educational Content**:
  - Literary language nuances
  - Cultural context requirements
  - Precise answer expectations
- **System Tradeoffs**:
  - Accuracy vs. latency decisions
  - Cloud vs. local inference costs
  - Chunk size optimization

3.4 Future Enhancements

The system could be improved through:

- **Bangla-Specific Optimizations**:
  - Custom pretrained embeddings
  - Domain-adapted tokenizer

  - Numeric normalization

- **Advanced Features**:
  - Follow-up question handling
  - User feedback integration

- **Production Readiness**:
  - Auto-scaling infrastructure
  - Continuous learning
  - Comprehensive monitoring

3.5 Final Remarks

This implementation demonstrates that effective bilingual RAG systems can be built for educational applications, even for lower-resource languages like Bangla. The project successfully balanced academic requirements with practical engineering considerations, delivering a system that:

- Respects cultural and linguistic nuances
- Maintains academic integrity through proper sourcing
- Provides immediate practical utility
- Serves as a foundation for future enhancements

The complete solution, including all source code and documentation, has been made publicly available on GitHub as required by the assessment guidelines.