# Generative AI Knowledge Base (Updated till Phase 3.7)

## 📌 Overview

This knowledge base documents the design, implementation, and learnings from the **Generative AI RAG Platform (AirNir)** across Phases 1 → 3.7. It consolidates architecture decisions, best practices, and technical lessons.

## 🔹 Phase 1 – Foundation

* Explored .NET AI use cases (chatbots, summarization, Q&A, code assist).
* Tested HuggingFace + Azure.
* Learned prompt engineering techniques:
  + Zero-Shot
  + Few-Shot
  + Role prompting

**Key Learning**: Establish baseline with prompt engineering before scaling with RAG.

## 🔹 Phase 2 – .NET Integration

* Built **.NET Core Web API** backend.
* SQL Server persistence layer.
* Modular JS frontend (chat, sessions, templates).
* Deployment on Azure (App Service + SQL Azure).
* Admin panel with CRUD for prompt templates (with versioning + preview).

**Key Learning**: A modular architecture allows prompt templates and logic to evolve without redeploying the core system.

## 🔹 Phase 3 – RAG Enhancements

### Phase 3.1 – Ingestion Layer

* Upload + chunking pipeline.
* Store documents + chunks in SQL.

### Phase 3.2 – Embedding + Vector DB

* Embeddings via OpenAI.
* pgvector integration in Postgres.
* Semantic retrieval queries.

### Phase 3.3 – Retrieval Service

* Semantic, keyword, and hybrid retrieval strategies.
* Debug UI for retrieval comparison.

### Phase 3.4 – Prompt Engineering

* Support for Zero-Shot, Few-Shot, Role, RAG, Hybrid.
* Admin UI for managing prompt styles.

### Phase 3.5 – RAG History + Docs Module

* Persist RAG runs in history.
* Filters: date, SLA, prompt style.
* Compare runs side-by-side.
* Export history.
* Added Docs module.

### Phase 3.6 – Analytics Dashboard

* SLA + latency monitoring.
* KPI metrics (SLA %, avg latency).
* Trends + usage charts.
* Drill-down navigation from analytics → history.

### Phase 3.7 – Advanced Analytics ✅

* **Multi-provider support**: OpenAI, Gemini, Claude.
* **SLA compliance tracking** per provider/model + prompt style.
* **Token tracking** (QueryTokens, ContextTokens, TotalTokens).
* Enhanced Analytics Service methods:
  + Provider/model aggregation.
  + SLA compliance.
  + Latency breakdowns.
  + Trends.
* **Frontend Updates**:
  + KPI widgets (Total Runs, Avg Latency, SLA %).
  + Extended charts (SLA by style, SLA by provider/model, latency trends).
* **Database Migration**:
  + Added token count fields to history table.

**Key Learning**: Token size directly impacts latency. Tracking tokens enables optimization (trim context, cap outputs, choose faster models).

## 📂 Project Structure (Phase 3.7)

/AirNir  
├── Library  
│ ├── ArNir.Core → Entities, DTOs, Config, Validations, Utils (Tokenizer)  
│ │ ├── DTOs/Analytics → AvgLatencyDto, SlaComplianceDto, ProviderAnalyticsDto, TrendDto, PromptStyleUsageDto  
│ │ └── Utils → TokenizerUtil.cs  
│ ├── ArNir.Data → DbContexts (SQL + Postgres), EF Migrations  
│ └── ArNir.Services → EmbeddingService, RetrievalService, RagService, RagHistoryService, Analytics  
│  
├── Presentation  
│ ├── ArNir.Admin → AdminLTE UI  
│ │ ├── Controllers → Docs, Retrieval, RAG Comparison, RAG History, Analytics  
│ │ ├── Views → Razor Views for RAG, History, Analytics  
│ │ ├── wwwroot/js → rag-comparison.js, rag-history.js, analytics.js  
│ │ └── Layout → Sidebar includes Analytics  
│ └── ArNir.Frontend → Planned end-user chat/search UI (Phase 3.8+)  
│  
├── sql → Migration scripts (Phase 3.7 adds token fields)  
└── docs → Architecture diagrams + documentation

## ✅ Outcomes Till Phase 3.7

* RAG pipeline is **production-ready** with:
  + Ingestion → Embeddings → Retrieval → Prompting → RAG Execution → History → Analytics.
* Supports **multiple providers/models**.
* Tracks **SLA compliance** and **latency**.
* Token insights available for deeper optimization.
* Analytics Dashboard with KPIs + charts + drill-down.

## ⏭ Next Steps (Phase 3.8)

* CSV/Excel export for analytics datasets.
* Token vs Latency analytics (scatter/line charts).
* Multi-select provider/model filters.
* Enhanced KPI widgets (tokens vs latency ratio).
* Configurable SLA thresholds per provider/model.