# 📘 GenerativeAI\_KnowledgeBase\_Phsase3.docx

This document tracks the progress of the Generative AI project across all phases, with focus on Phase 3 developments.

## 🔹 Phase 1 – Foundation

* Explored use cases in .NET apps (chatbots, Q&A bots, summarization, content generation, code assist).
* Tested HuggingFace models for API demo + Azure deployment.
* Learned Prompt Engineering basics: Zero-Shot, Few-Shot, Role Prompting.
* Findings:
  + Few-shot & role prompting gave better results.
  + Zero-shot was verbose/unreliable.

## 🔹 Phase 2 – .NET Integration

### **Phase 2.1 – Backend Integration**

* .NET Core Web API with ChatController endpoints (send, stream, history, sessions, duplicate-session, delete).
* Services layer with IOpenAiService + IChatHistoryService.
* Persistence in SQL Server (ChatSessions, ChatMessages).

### **Phase 2.2 – Frontend Integration**

* Modularized JS files: chat.js, sessions.js, templates.js, utils.js, main.js.
* Features: streaming, typing dots, session sidebar, model selector, prompt preview.

### **Phase 2.3 – Deployment**

* Azure App Service + SQL Azure.
* Fixed CORS + connection strings in App Config.

### **Phase 2.4 – Prompt Templates & Clean UI**

* Templates stored in DB with parameters (tone, length).
* buildPrompt() inserts parameters into templates.
* buildPromptPreview() updates preview instantly.
* Admin panel for template CRUD.

## 🔹 Phase 3 – RAG Enhancements

### **Phase 3.1 – Architecture Foundations**

* Established modular architecture for RAG services.
* Integrated retrieval pipelines and OpenAI service abstraction.
* Defined DTOs for retrieval results and responses.

### **Phase 3.2 – RAG Service Integration**

* Implemented retrieval service with hybrid search support.
* Added OpenAI completion integration with prompt building.
* Standardized DTOs for consistent data flow.

### **Phase 3.3 – Admin UI for Comparisons**

* Created AdminLTE-based interface for running RAG comparisons.
* Supported side-by-side comparisons of multiple providers/models.
* Added comparison history view with details modal.

### **Phase 3.4 – Prompt Engineering**

* Introduced **advanced prompt engineering techniques**:
  + Zero-Shot
  + Few-Shot
  + Role Prompting
  + RAG-Augmented
  + Hybrid Role + RAG
* Integrated **dynamic prompt generation** into the RAG service.
* Extended DB schema with PromptStyle column to persist prompt type used per run.
* Prompt experimentation enabled in Admin UI.

### **Phase 3.5 – RAG History Enhancements**

* Extended **RAG Comparison Page**:
  + Query + PromptStyle selector.
  + Results: Baseline vs RAG answers.
  + Retrieved context with expandable preview.
  + SLA badge (✅ OK / ⚠️ Slow).
* Extended **History Page**:
  + Filters: SLA, date range, query, PromptStyle.
  + Details modal with chunks.
  + Compare Mode: select multiple runs and compare side-by-side.
  + Export: CSV/Excel (single/multiple runs, with chunks).
* Added **Docs Module**:
  + Upload new documents
  + Edit/Delete existing documents
  + Rebuild embeddings on demand
* Migrated UI to **Bootstrap 5** (Run button, spinners, toggles).

## 🔹 Updated Project Structure (till Phase 3.5)

/ArNir  
├── Library  
│ ├── ArNir.Core → Entities, DTOs, Config, Validations  
│ ├── ArNir.Data → DbContexts (SQL Server + Postgres), EF Core migrations (separate SqlServer/Postgres folders)  
│ └── ArNir.Services → Business logic (EmbeddingService, RetrievalService, RagService, RagHistoryService)  
│  
├── Presentation  
│ ├── ArNir.Admin → AdminLTE UI (ASP.NET Core MVC project)  
│ │ ├── Views  
│ │ ├── wwwroot/js  
│ │ ├── Controllers  
│ │ │ ├── Docs (Upload/Edit/Delete, Rebuild Embeddings)  
│ │ │ ├── Embedding Test Page (/Embedding/Test)  
│ │ │ ├── Retrieval Test Page (/Retrieval/Test)  
│ │ │ ├── RAG Comparison Page (/RagComparison)  
│ │ │ └── RAG History Page (/RagHistory)  
│ └── ArNir.Frontend → End-user search/chat interface (planned Phase 3.6)  
│  
├── sql  
│ ├── create\_tables.sql  
│ ├── update\_documents\_chunks.sql  
│ ├── update\_embeddings.sql  
│ └── update\_rag\_history.sql  
│  
└── docs  
 ├── Phase3  
 │ ├── Phase3\_RAG\_Architecture.png  
 │ ├── Phase3.3\_Architecture.png  
 │ ├── Phase3.4\_Architecture.png  
 │ ├── Phase3.5\_Architecture.png  
 │ ├── Phase3.5\_Technical\_Architecture.png  
 │ ├── Phase3.4\_RAG.md  
 │ └── Phase\_3\_RAG.md  
 ├── GenerativeAI\_KnowledgeBase.md  
 ├── GenerativeAI\_KnowledgeBase\_Phsase3.docx  
 └── README.md

## 🔹 Updated Architecture Diagrams

**Phase 3.5 Technical Architecture:** Phase 3.5 Technical Architecture

**Phase 3.5 System Architecture:** Phase 3.5 Architecture

## ✅ Final Phase 3 Achievements

* **Robust RAG pipeline** with hybrid retrieval and OpenAI integration.
* **Prompt Engineering** embedded and logged in DB.
* **Admin UI** supports running, reviewing, and comparing experiments.
* **RAG History** with filters, comparisons, exports.
* **Docs Module** for document lifecycle (Upload/Edit/Delete/Rebuild).
* **Database schema** extended for PromptStyle and embeddings.
* **Bootstrap 5 migration** for smoother UI.

👉 Ready for **Phase 3.6 – Analytics Kickoff**, where focus will shift to visualization of results and performance trends.