AI Health Platform – Combined Technical & Presentation Documentation

1. Introduction

This document merges the system documentation with the content provided in the presentation to form a unified, comprehensive technical report for **Tabibak – AI-Powered Proactive Healthcare**, covering both patient-facing and doctor-facing methodologies.

The platform leverages AI to: - Provide instant medical guidance for patients. - Support doctors with evidence-based RAG medical search. - Integrate Computer Vision + NLP into one smart ecosystem.

---

2. System Overview

2.1 Users & Stakeholders

- **Patients** – seeking medical advice, symptom clarification, image interpretation.
- **Doctors** – requiring fast literature-backed answers & research assistance.
- **Healthcare Providers** – using automated triage and decision support.

2.2 Core Objectives

- Develop AI systems with human-like interaction.
- Provide reliable, evidence-based medical information.
- Create a unified platform that combines multiple AI technologies.
- Validate with real-world performance & evaluation metrics.

---

3. Patient Chatbot – Arabic Medical Assistant

3.1 Architecture Philosophy

The Patient Chatbot module is not merely a fine-tuned model; it is a **hybrid architectural system** designed to overcome the limitations of standard LLMs in medical contexts. It addresses specific challenges such as "Catastrophic Forgetting" (losing general language ability while learning medicine) and "Hallucination" through a custom-engineered architecture that separates general reasoning from medical specialization.

3.2 Data Collection

Dataset: **Shifaa Arabic Medical Consultations**

- 16 CSV files (medical specialties).

- 84,422 real question–answer pairs.

- Columns: Question Title, Question, Answer, Doctor, Diagnosis, Date.

3.3 Data Preprocessing

Before feeding data into the model, a rigorous cleaning pipeline was applied to maximize token efficiency and semantic focus:

- Extracted only **Question** & **Answer**.
- **Noise Reduction:** Systematic removal of non-clinical text using RegEx.
    - *Target:* Religious salutations (e.g., "السلام عليكم", "جزاك الله خيرا"), polite introductions, and doctor signatures.
    - *Rationale:* These tokens consume context window space without adding medical value. Removing them forces the model to focus immediately on the symptom-diagnosis relationship.
- **Instruction Formatting:**
    - Data was restructured into a JSONL format compatible with instruction-tuning:

```
{
  "instruction": "Patient Question (Symptoms/History)",
  "output": "Doctor Answer (Diagnosis/Advice)"
}
```

- **Stratified Splitting:**
    - The dataset was split using a seeded random state (42) to ensure reproducibility:
        - **Train:** 80% (Learning patterns).
        - **Validation:** 10% (Tuning hyperparameters).
        - **Test:** 10% (Final unseen evaluation).

3.4 Model Selection & Training

Models used:

- Meta-Llama-3.1-8B-Instruct

- Unsloth Llama-3.1-8B

- Qwen 2.5-7B-Instruct (custom medical adapter)

Fine-tuning:

-we leveraged **Unsloth**:

- **Kernel Optimization:** Unsloth manually derives backpropagation gradients, making training **2x faster** and reducing VRAM usage by **60%**.
- **4-Bit Quantization (NF4):**
    - The model handles weights in 4-bit Normal Float (NF4) format rather than 16-bit.
    - **Double Quantization:** Used to quantize the quantization constants themselves, squeezing every bit of memory efficiency without significant accuracy loss.

- Sequence length: 1024 tokens

- LoRA rank: R = 64 - LR = 2e-4

3.5 Custom Medical Adapter (Qwen Only)

This is the core technical differentiator of the project. Instead of relying solely on LoRA, we injected a brand-new neural network layer into the pre-trained model.

A. The "Bottleneck" Architecture

The adapter acts as a filter that specializes in medical features.

- **Input Dimension:** 3584 (Hidden size of Qwen 7B).
- **Down-Projection:** Linear Layer compresses data to **512 dimensions**. This "bottleneck" forces the layer to discard noise and retain only the most critical medical features.
- **Activation Function:** GELU (Gaussian Error Linear Unit) allows for complex, non-linear mapping of medical concepts.
- **Regularization:** Dropout(0.05) randomly zeros out neurons during training to prevent the model from memorizing specific answers (Overfitting).
- **Up-Projection:** Linear Layer expands back to 3584 dimensions to match the model's flow.

B. Dynamic Injection (Monkey Patching)

The adapter was integrated using a Python technique . We intercepted the model's standard forward() function and replaced it with a custom logic:

```python
class MedicalAdapterLayer(nn.Module):

    def __init__(self, hidden_size=3584, bottleneck_size=512):
        super().__init__()
        self.medical_projection = nn.Sequential(
            nn.Linear(hidden_size, bottleneck_size),
            nn.GELU(),
            nn.Dropout(0.05),
            nn.Linear(bottleneck_size, hidden_size),
            nn.LayerNorm(hidden_size)
        )

    def forward(self, hidden_states):
        return hidden_states + 0.1 * self.medical_projection(hidden_states)
```

```python
def forward_with_medical_adapter(self, input_ids=None, attention_mask=None, **kwargs):
    kwargs['output_hidden_states'] = True
    outputs = original_model_forward(self, input_ids=input_ids, attention_mask=attention_mask, **kwargs)

    if hasattr(outputs, 'hidden_states') and outputs.hidden_states is not None and len(outputs.hidden_states) > 0:
        last_hidden = outputs.hidden_states[-1]

        with torch.cuda.amp.autocast(enabled=False):
            adapted = medical_adapter(last_hidden.float())

        outputs.hidden_states = outputs.hidden_states[:-1] + (adapted,)

    return outputs
```

- **Residual Connection:** The term (0.1 * adapter_output) means the medical knowledge acts as a "nudge" (10% influence) rather than a replacement. This ensures the model speaks fluent Arabic (original knowledge) while gaining medical expertise (adapter knowledge).

3.6 Hyperparameter Optimization: Ant Colony Algorithm (ACO)

Standard training uses fixed parameters (e.g., Temperature = 0.7). We treated parameter selection as a **search path problem**.

A. Algorithm Logic

1. **The Environment:** A 3D search space consisting of Temperature (Creativity), Top_p (Vocabulary Restriction), and Max_tokens.
2. **The "Ants":** 5 parallel agents per iteration choose random parameter combinations.
3. **Pheromone Trails:**
    - If an Ant's combination results in a high **BERTScore** on the validation set, it leaves a strong "digital pheromone."

    - In the next iteration, new Ants are statistically more likely to choose parameters near the strong pheromones.

4. **Evaporation:** Pheromones evaporate by 70% each cycle to prevent getting stuck in local optima (encouraging exploration).

B. Optimization Outcome

- **Iterations:** 4 Cycles.
- **Winning Configuration:**
    - Temperature: **0.7** (Balanced creativity).
    - Top_p: **0.99** (Broad vocabulary access).
    - This configuration yielded a **BERTScore F1 of 69.85%**, significantly outperforming random grid search.

3.6 Evaluation Framework

Traditional metrics like Accuracy don't work for generative chat. We used a multi-metric approach:

1. **BERTScore (Primary Metric):**
    - Uses a pre-trained Arabic BERT model to compare the *semantic embedding* of the generated answer vs. the doctor's real answer.
    - *Why?* It detects that "Headache" and "Cephalalgia" are the same, whereas standard metrics would mark them as errors.
2. **Exact Match (EM):** Used strictly for checking if specific medical entities (e.g., drug names) were recalled correctly.
3. **Human-Aligned Testing:** The test set (10% of data) was completely unseen during training to simulate real-world patient interactions.

---

4. Doctor Chatbot – English RAG System for PubMed

Overview

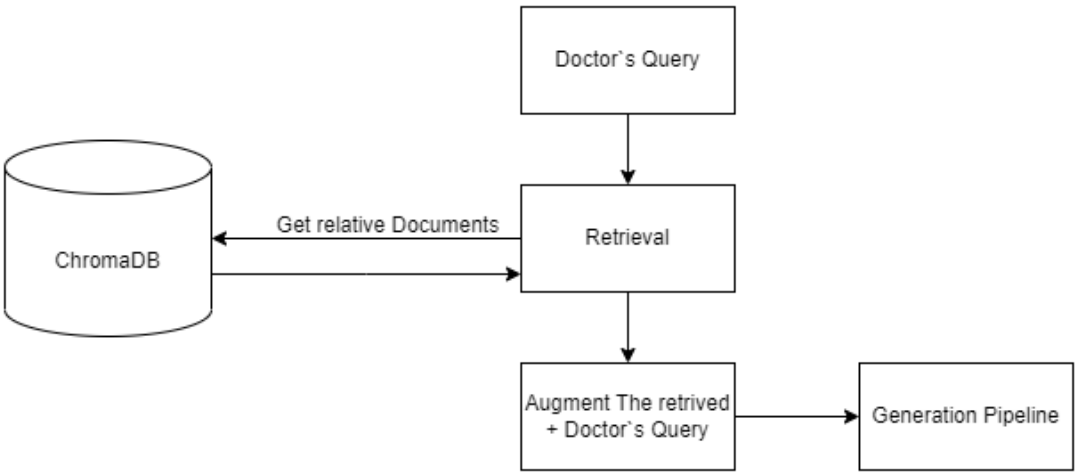
The **Doctor Chatbot** is an English-language, Retrieval-Augmented Generation (RAG)–based assistant designed to support medical professionals with **evidence-based, literature-backed answers** derived from PubMed and other trusted sources.

It integrates natural language understanding, dense and sparse document retrieval, and generative reasoning using **Google Gemini** while maintaining strict factual grounding.

Core Objectives

- Provide **evidence-based answers** to physicians' clinical questions in English.
- Retrieve relevant information from **PubMed medical literature**.
- Generate structured, transparent answers with **chain-of-thought explanations**.
- Ensure high retrieval quality and factual correctness using **Hybrid RAG Evaluation**.

System Architecture



Main Components

The Doctor Chatbot pipeline consists of the following modules:

1. **Data Preparation**
    - Fetches and cleans PubMed abstracts.
    - Removes duplicates, citations, and non-English entries.
2. **Chunking**
    - Recursively splits long abstracts into smaller, semantically coherent chunks.

3. **Vector Databases**
    - **ChromaDB (Dense Semantic Indexing)** using MedEmbed-large-v0.1.
    - **BM25 (Lexical Indexing)** using tokenized text.
    - Combined via α-weighted hybrid retrieval.
4. **Generation Pipeline**
    - Two-stage prompt to ensure reliability:
        1. **Reasoning Process** (document analysis & synthesis)
        2. **Structured Clinical Answer**
5. **Evaluation Framework**
    - Computes **Precision@K**, **Recall@K**, **MRR**, **MAP**, and **Hybrid Comparisons**.
6. **Ground Truth Builder**
    - Uses **Groq LLMs** for automatic dataset creation and relevance scoring.

---

1. Indexing & Preprocessing Pipeline

PubMedDataCleaner

Purpose: Retrieve and clean PubMed articles.

Features:

- Fetches using **Entrez API**.
- Cleans abstracts (removes author info, DOIs, references).
- Filters only **English-language** texts.
- Removes noise, citations, and publisher information.
- Saves results as JSON for downstream processing.

Output Fields:

- pmid
- title
- abstract

- source

---

2. Document Chunking

PubMedChunker

Parameters:

- chunk_size: 1000 characters
- chunk_overlap: 200 characters

Function:
Splits abstracts into logical chunks using LangChain's RecursiveCharacterTextSplitter, ensuring minimal semantic loss.

Output Format:

```json
{
  "text": "Title: ... Abstract: ...",
  "metadata": {
      "pmid": "12345",
      "title": "Study on ...",
      "source": "Journal of Medicine"
  }
}
```

---

3. Vector Database

3.1 Dense Vector Store – ChromaVectorStore

- Embedding Model: abhinand/MedEmbed-large-v0.1.
- Persistent storage in /chroma_db.
- Supports:
    - .build_vector_store()
    - .retrieve(query, k)
    - .similarity_search_with_score()

3.2 Hybrid Vector Store – HybridChromaVectorStore

Combines semantic and lexical retrieval for optimal performance.

Fusion Formula:

HybridScore = α * DenseScore + (1 - α) * SparseScore

Features:

- Builds **ChromaDB** (dense embeddings).
- Builds **BM25** index (token-based).
- Supports direct or hybrid retrieval via:
    - .retrieve() → dense only
    - .hybrid_retrieve() → fused retrieval

Advantages:

✅ Combines meaning + keyword overlap
✅ Handles short queries better
✅ Enhances recall and precision simultaneously

---

4. Generation Pipeline (RAG System)

Implementations:

- **MedicalRAGGenerationPipeline** (Dense RAG)
- **HybridMedicalRAGPipeline** (Hybrid RAG)

Steps:

1. **Retrieval Stage**
    - Retrieves top-K documents using either dense or hybrid retrieval.
2. **Prompt Construction**
    - Builds a structured two-stage prompt:
        - **Stage 1:** Document analysis, evidence extraction, synthesis.
        - **Stage 2:** Structured answer with sources, confidence, and caveats.
3. **Generation Stage**

- Utilizes **Google Gemini API** (gemini-2.5-flash-lite-preview-06-17).
- Restrictive prompt ensures no hallucinations or unverifiable claims.

4. **Output Contains:**
    - answer: Evidence-based response.
    - sources_used: Documents referenced.

---

5. Evaluation Framework

5.1 Dense Evaluator – RAGRetrievalEvaluator

Calculates:

- Precision@K
- Recall@K
- MRR
- AP / MAP


```
📊 STATISTICAL SUMMARY:
--------------------------------------------------------------------------------
Average Precision (AP) Statistics:
  Mean:   0.3957
  Median: 0.3545
  Std:    0.1372
  Min:    0.1940
  Max:    1.0000

Mean Reciprocal Rank (MRR) Statistics:
  Mean:   1.0000
  Median: 1.0000
  Std:    0.0000
```

5.2 Hybrid Evaluator

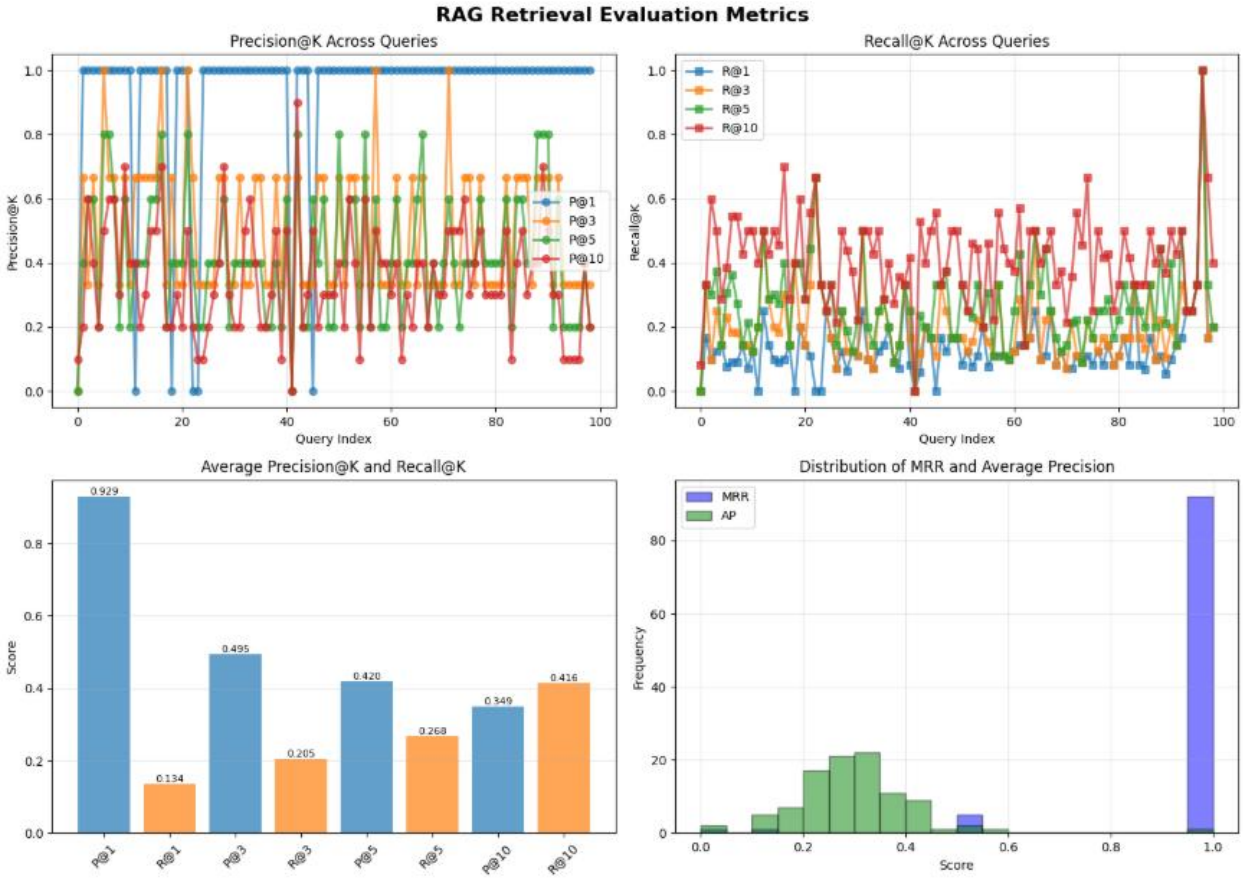
Enhanced with capability to compare **Dense**, **Sparse**, and **Hybrid** systems.

Core Metrics:

- Precision@K, Recall@K
- MRR, MAP
- Retrieval time (latency)
- Optional NDCG support

Comparison Methodology:

- Tests Dense, Sparse, Hybrid (α = 0.3 / 0.5 / 0.7).
- Generates visual comparisons (heatmap, bar charts).



```
📊 STATISTICAL SUMMARY:
--------------------------------------------------------------------------------

AVERAGE_PRECISION:
  Mean:   0.2974
  Median: 0.2851
  Std:    0.1222
  Min:    0.0000
  Max:    1.0000

MRR:
  Mean:   0.9558
  Median: 1.0000
  Std:    0.1697
  Min:    0.0000
  Max:    1.0000

PRECISION@5:
  Mean:   0.4202
  Median: 0.4000
  Std:    0.2030
  Min:    0.0000
  Max:    0.8000

RECALL@5:
  Mean:   0.2681
  Median: 0.2500
  Std:    0.1362
  Min:    0.0000
  Max:    1.0000
```

```
================================================================================
📊 COMPARISON SUMMARY TABLE
================================================================================

          method      map      mrr  precision@5  recall@5  avg_retrieval_time
           dense 0.389219 1.000000     0.428283  0.272844            0.033234
          sparse 0.184316 0.876347     0.274747  0.183844            0.408136
hybrid_alpha_0.3 0.326591 0.959091     0.373737  0.242836            0.496232
hybrid_alpha_0.5 0.354042 0.969697     0.385859  0.250027            0.492661
hybrid_alpha_0.7 0.365409 0.979798     0.406061  0.261082            0.480381

================================================================================
🏆 BEST PERFORMERS
================================================================================
MAP                 : dense                          (0.3892)
MRR                 : dense                          (1.0000)
PRECISION@5         : dense                          (0.4283)
RECALL@5            : dense                          (0.2728)
FASTEST             : dense                          (0.033s)
================================================================================
```

---

6. Ground Truth Generation

GroundTruthBuilderGroq

Creates supervised evaluation datasets using multiple models via Groq API.

Features:

- Automatic query generation from documents.
- Document relevance judgment (score 0–1).

to PDF/Word - Add more technical deep dives