Université Internationale de Rabat

THE INNOVATIVE UNIVERSITY

A Safety-Oriented Artificial Intelligence System for Medical Triage and Health Information Assistance

Prepared by:

ILEGH SALMA

KASSIF RANIYA

ELHAIRECH AMINE

SEHLI NASSIM

Prepared by:

MR GAMOUH HAMZA

MR HAFIDI HAKIM

Abstract

Artificial intelligence has the potential to improve access to medical information and support early decision-making in healthcare systems. However, the use of AI in medical triage requires strict safety constraints, explainable reasoning, and robust system design. This project presents the design and implementation of an AI-powered medical triage assistant that combines natural language processing, semantic knowledge retrieval, and structured medical reasoning within a single hybrid architecture.

The system operates in two complementary modes. The first mode performs medical triage by analyzing free-text symptom descriptions and classifying case urgency using a conservative safety-first approach. Experimental testing demonstrated that all triage outputs were classified as either MODERATE or CRITICAL, reflecting intentional risk aversion in the presence of incomplete or ambiguous user input. The second mode functions as a medical chatbot, providing educational and non-diagnostic medical information while explicitly avoiding diagnoses or treatment recommendations.

The proposed architecture integrates a locally deployed large language model via Ollama for natural language understanding, a FAISS vector database for semantic retrieval of medical knowledge, and a Neo4j knowledge graph that models explicit symptom–clinical pattern–disease relationships. Confidence scores and urgency levels are computed using weighted graph-based reasoning rather than being inferred directly by the language model, ensuring transparency and reproducibility. The system includes fallback mechanisms that allow continued operation in the event of partial component failures.

In addition, a monitoring module records performance metrics such as latency, error rates, and urgency distribution, enabling objective evaluation of system behavior. The results demonstrate that a hybrid AI architecture can support safe, explainable, and robust medical triage while maintaining clear ethical boundaries. Although the system is not intended for clinical diagnosis, it provides a strong academic prototype for safety-oriented AI decision support in healthcare applications.

Table of Contents

Table of Illustrations

Introduction

Artificial intelligence (AI) has become an essential tool in modern healthcare, particularly in areas that require fast decision-making and accessible medical information. Two important challenges in this domain are medical triage, where symptoms must be assessed to determine urgency, and patient education, where users seek understandable and safe medical explanations. Addressing both challenges in a single system requires a careful balance between accuracy, safety, and usability.

This project presents the design and implementation of an AI-Powered Medical Triage Assistant, a hybrid system that operates in two complementary modes.
The first mode is a medical triage module, which analyzes free-text symptom descriptions provided by users and classifies the urgency of their condition (CRITICAL or MODERATE, depending on assessed risk). The second mode is a medical chatbot, which allows users to ask general health-related questions and receive educational, non-diagnostic responses in natural language.

To achieve this functionality, the system integrates multiple AI and data technologies. A Large Language Model (LLM) running locally via Ollama is responsible for understanding natural language input and generating structured medical reasoning. A FAISS vector database enables semantic retrieval of relevant medical knowledge, while a Neo4j graph database models explicit relationships between symptoms, clinical patterns, and diseases. This hybrid architecture allows the system to combine flexible language understanding with structured medical reasoning.

Patient safety is a core design principle of the system. The chatbot is explicitly constrained to provide informational guidance only, without diagnosing conditions or prescribing treatments. Similarly, the triage module is designed to prioritize caution and encourage professional medical consultation whenever uncertainty exists.

In addition to its analytical and conversational capabilities, the system includes a monitoring and metrics module that tracks performance indicators such as response latency, success rate, and error rate. A modern web-based user interface provides users with access to symptom analysis, medical questions, historical results, and system metrics in a clear and intuitive manner.

Overall, this project demonstrates how a multi-layer AI system can be applied responsibly to healthcare-related applications by combining triage decision support, conversational medical information, and continuous system monitoring.

Chapter 1: Problem Statement & Objectives

1. Problem Statement

During the experimental phase of this project, all tested symptom inputs resulted in an urgency level classified as **MODERATE** or **CRITICAL**. This behavior is not a flaw of the system; rather, it reveals a fundamental challenge in medical AI systems: **ambiguity and risk aversion in symptom interpretation**.

In real-world scenarios, users rarely describe symptoms precisely or completely. Even apparently benign complaints (such as fatigue or headache) may hide more serious conditions when duration, intensity, or associated symptoms are unknown. As a result, a medical triage system must be **conservative by design**, prioritizing patient safety over reassurance.

The core problems addressed by this project are therefore:

- **Medical uncertainty**: Limited or vague inputs make low-risk classification unsafe.
- **Risk of under-triage**: Incorrectly classifying a serious condition as LOW urgency can have severe consequences.
- **Limitations of purely statistical AI**: Large language models alone may underestimate severity or hallucinate confidence.
- **Need for structured reasoning**: Symptoms must be interpreted through medically meaningful relationships rather than isolated keywords.

The observed predominance of MODERATE and CRITICAL outcomes during testing reflects the system's **intentional bias toward safety**, which is a critical requirement in medical triage contexts.

2. Project Objectives

Taking these constraints into account, the project was designed with objectives that explicitly favor **safe decision-making** over optimistic classification.

The main objectives are :

- **Provide conservative medical triage**
  - Classify urgency levels with a safety-first approach
  - Prefer MODERATE over LOW when information is incomplete
  - Escalate to CRITICAL when symptom combinations match high-risk clinical patterns
- **Support two complementary interaction modes**
  - **Triage mode**: Structured urgency assessment with confidence scoring
  - **Chatbot mode**: Educational medical dialogue without urgency classification
- **Combine multiple reasoning layers**
  - Ollama (LLM) for natural language understanding and synthesis
  - FAISS for semantic similarity with medical texts
  - Neo4j for explicit symptom → clinical pattern → disease relationships
- **Ensure robustness during failures**

    - Maintain functional triage even when AI or graph components are unavailable
    - Fall back to conservative defaults when uncertainty is high
- **Expose system behavior through monitoring**
    - Log urgency distributions
    - Measure latency and error rates
    - Detect abnormal response patterns

Chapter 2 : System Architecture

The medical triage assistant is built on a **modular, service-oriented architecture** designed to ensure reliability, scalability, and fault tolerance. Rather than relying on a single AI model, the system combines **multiple complementary components**, each responsible for a specific role in the decision-making pipeline.

This architecture allows the system to continue operating even if one component becomes unavailable (for example, if the language model is offline), which is essential in safety-critical medical applications.

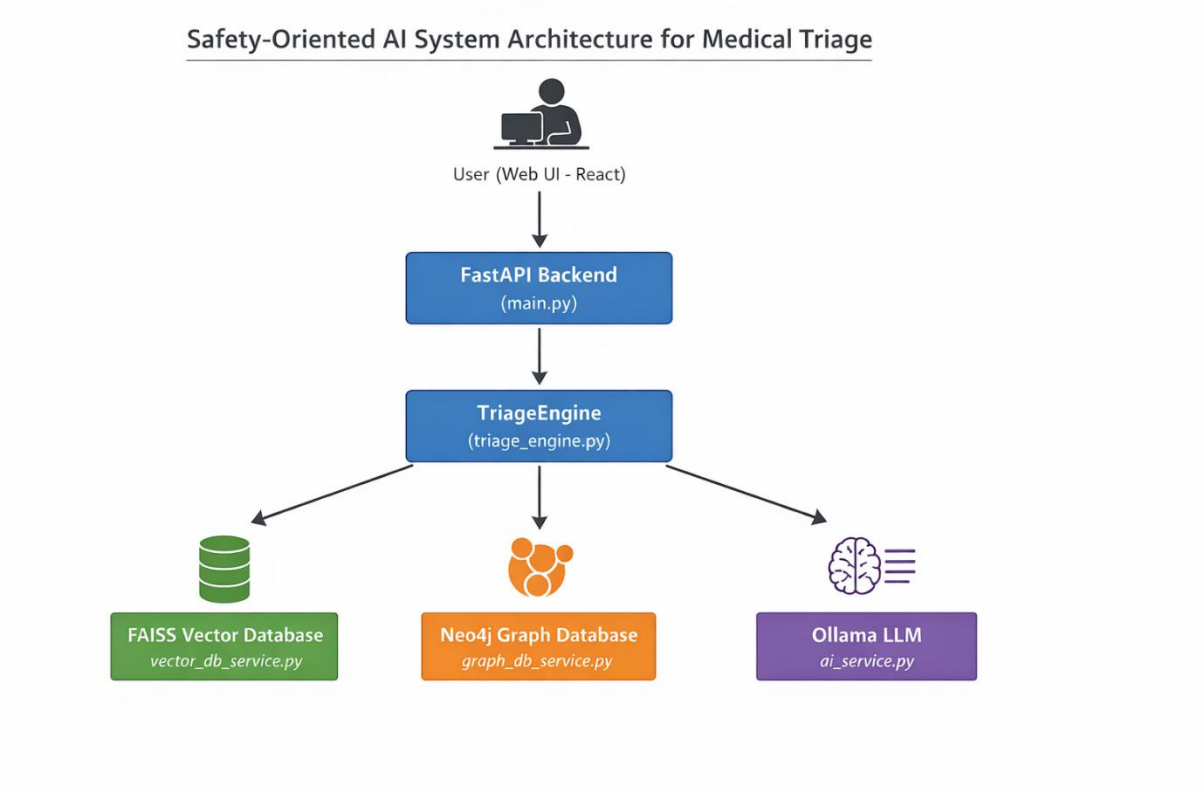


Figure 1:Architecture of the Safety-Oriented AI System for Medical Triage and Health Information Assistance.

1. High-Level Architecture Overview

At a high level, the system is composed of five main layers:

- User Interface (Frontend)
- API & Orchestration Layer (FastAPI Backend)
- AI Reasoning Layer (Ollama LLM)
- Knowledge Retrieval Layer (FAISS Vector Database)
- Structured Medical Reasoning Layer (Neo4j Knowledge Graph)

Each layer communicates through well-defined interfaces, ensuring separation of concerns and easier maintenance.

2. Frontend Layer

The frontend provides a **chat-based user interface** that allows users to:

- Enter symptom descriptions in natural language
- Interact with the medical chatbot
- Receive triage results (urgency level, confidence, advice)

Key characteristics:

- Focused on usability and clarity
- No medical logic is implemented on the client side
- All decisions are delegated to the backend

The frontend acts purely as an **interaction layer**, ensuring that medical reasoning remains centralized and controlled.

3. Backend & API Layer (FastAPI)

The backend is implemented using **FastAPI**, serving as the **central orchestrator** of the system. Its responsibilities include:

- Receiving and validating user requests
- Routing requests to the appropriate services
- Aggregating results from multiple components
- Returning structured and safe responses

Main API endpoints:

- /triage → Medical urgency assessment
- /chat → Educational medical chatbot
- /metrics → System monitoring and performance data
- /history → Anonymized triage history

FastAPI was chosen for its:

- High performance
- Built-in validation (Pydantic)
- Clear separation between request handling and business logic

4. AI Reasoning Layer (Ollama)

The AI reasoning layer uses **Ollama** to run a local large language model (LLM). This component is responsible for:

- Understanding free-text symptom descriptions
- Generating structured triage responses
- Producing educational chatbot answers

Important architectural decisions:

- The LLM **does not work alone**
- It receives contextual information from FAISS and Neo4j
- It is constrained to output structured JSON for triage

If the AI model becomes unavailable, the system automatically falls back to **graph-based reasoning**, ensuring continuity.

5. Knowledge Retrieval Layer (FAISS Vector Database)

FAISS is used to store and retrieve medical knowledge documents through semantic similarity. Its role in the architecture:

- Retrieve relevant medical context based on user input
- Enrich the AI prompt with trustworthy background information
- Improve response relevance without increasing hallucination risk

FAISS operates only when the input contains sufficient semantic content, reducing unnecessary computation and latency.

6. Structured Medical Reasoning Layer (Neo4j)

Neo4j is used to model a **medical knowledge graph** with explicit relationships:

- Symptom → ClinicalPattern → Disease

This layer provides:

- Deterministic reasoning
- Explainable symptom–disease associations
- Weighted confidence propagation

Unlike the language model, Neo4j ensures:

- No hallucinations
- Transparent reasoning paths
- Reproducible results

The graph plays a crucial role in fallback scenarios and in validating AI-generated insights.

7. Fault Tolerance & Graceful Degradation

A core architectural principle of the system is graceful degradation:

- If Ollama fails → Use Neo4j-based triage
- If Neo4j fails → Use AI + vector context
- If all AI components fail → Return a safe default response

This ensures that the system never crashes silently and always prioritizes patient safety

Chapter 3: Data Sources & Knowledge Representation

The effectiveness of a medical triage system depends largely on the quality, structure, and diversity of its data sources. In this project, no single data source is considered sufficient on its own. Instead, the system relies on a hybrid knowledge strategy, combining unstructured medical text, vectorized semantic knowledge, and a structured medical graph.

This approach allows the system to answer a wide variety of inputs while maintaining reliability and safety.

1. Unstructured Medical Knowledge

Unstructured data represents general medical knowledge expressed in natural language. This type of data is essential for understanding broad symptom descriptions and contextual medical information.

In the system, unstructured knowledge includes:

- Short medical explanations (e.g., fatigue, fever, chest pain)
- Educational health content
- General triage guidelines

This data is not queried directly. Instead, it is processed and embedded into numerical vectors to enable semantic search.

2. Vectorized Knowledge Representation (FAISS)

To handle unstructured medical text efficiently, the project uses FAISS as a vector database.

Role of FAISS

FAISS is responsible for:

- Storing vector embeddings of medical knowledge documents
- Retrieving semantically relevant information based on user input
- Providing contextual background to the AI model

Rather than matching exact keywords, FAISS allows the system to understand meaning, enabling responses even when the user uses informal or incomplete language.

Why FAISS Was Chosen

- Extremely fast similarity search
- Lightweight and local (no external API dependency)
- Suitable for small to medium-sized medical datasets

FAISS is activated only when the user input contains sufficient semantic richness, which helps reduce unnecessary computation and response time.

3. Structured Medical Knowledge (Neo4j Graph)

While vector search is powerful, it lacks explicit medical structure. To address this, the system uses Neo4j to represent medical knowledge in a graph format.

Graph Structure

The knowledge graph follows a clear and interpretable structure:

- **Symptom** nodes (e.g., *Chest Pain, Fever*)
- **ClinicalPattern** nodes (e.g., *Cardiac Ischemic Pattern*)
- **Disease** nodes (e.g., *Heart Attack*)

Relationships include:

- (:Symptom)-[:PART_OF {weight}]->(:ClinicalPattern)
- (:ClinicalPattern)-[:INDICATES {confidence}]->(:Disease)

Each relationship carries a weight or confidence score, allowing probabilistic reasoning.

4. Knowledge Encoding & Confidence Attribution

Medical certainty is not binary. For this reason, the system encodes **confidence values** directly into the knowledge representation.

- weight reflects how strongly a symptom contributes to a clinical pattern
- confidence reflects how strongly a pattern suggests a disease

These values are :

- Defined manually based on medical plausibility
- Used mathematically during inference
- Combined dynamically at runtime

This design allows the system to distinguish between weak signals and strong clinical indicators.

5. Hybrid Knowledge Usage Strategy

Each knowledge source plays a specific role during inference:

- **FAISS** → semantic context & background knowledge
- **Neo4j** → deterministic and explainable reasoning
- **AI model** → natural language understanding and synthesis

Rather than competing, these sources complement each other, resulting in more robust and consistent triage outcomes.

6. Limitations of the Knowledge Base

Despite the hybrid approach, certain limitations remain:

- The medical graph does not cover all possible diseases
- FAISS knowledge depends on the quality of embedded documents
- Confidence scores are approximations, not clinical certainties

These limitations are acknowledged by design, and the system explicitly avoids claiming medical diagnosis

Chapter 4: AI Models & Reasoning Logic

The intelligence of the system does not rely on a single model or technique. Instead, it is built on a layered reasoning strategy, where each component contributes a specific type of intelligence. This design avoids the weaknesses of purely generative models and improves reliability in a medical context.

The system combines rule-based reasoning, graph-based inference, vector similarity search, and large language model (LLM) synthesis.

1. Role of the Large Language Model (Ollama)

The Large Language Model (LLM), deployed locally using Ollama, serves as the system's natural language understanding and generation engine.

Its primary responsibilities include:

- Understanding free-text symptom descriptions
- Handling vague, incomplete, or conversational inputs
- Generating human-readable medical advice
- Producing structured JSON outputs for triage results

The LLM **does not diagnose diseases**. Instead, it interprets context and synthesizes insights provided by other components.

Model Choice

- The system uses a **quantized instruction-tuned model** (llama3:8b-instruct-q4)
- Optimized for low latency and local execution
- Ensures data privacy (no cloud calls)

2. Prompt Engineering & Output Control

To guarantee safe and predictable behavior, the LLM is constrained using strict prompt engineering.

Key constraints include:

- Mandatory JSON-only output for triage
- Fixed urgency categories (CRITICAL, MODERATE)
- Explicit prohibition of diagnoses or prescriptions
- Safety-first instructions in ambiguous cases

This approach reduces hallucinations and ensures compatibility with downstream system logic.

3. Graph-Based Medical Reasoning (Neo4j)

Neo4j provides the system with explicit medical reasoning capabilities that LLMs alone cannot guarantee.

The reasoning process follows a deterministic path:

1. Extract symptom keywords from user input
2. Match symptoms to clinical patterns

3. Infer potential diseases through weighted relationships

Each inference step is **traceable and explainable**, which is essential for medical decision support systems.

4. Confidence Computation Logic

Confidence scores are not generated arbitrarily by the AI. Instead, they are computed mathematically using graph data.

The base confidence is calculated as:

- Symptom-to-pattern weight × Pattern-to-disease confidence
- Aggregated across matched symptoms
- Averaged to avoid bias from single strong signals

An adaptive urgency score is then computed:

- **Confidence × (1 + 0.15 × number of matched symptoms)**

This mechanism explains why the same symptom set may result in:

- MODERATE urgency in weak combinations
- CRITICAL urgency when multiple strong indicators are present

5. Vector Similarity Reasoning (FAISS)

FAISS plays a complementary role by enabling **semantic medical reasoning**.

Its responsibilities include:

- Identifying medically relevant concepts even when wording differs
- Providing background medical context to the LLM
- Supporting long, descriptive symptom inputs

FAISS is selectively activated to:

- Reduce latency
- Avoid unnecessary computation on short inputs

6. Hybrid Decision Strategy

The system applies a priority-based reasoning order:

1. **Graph reasoning** (most reliable, explainable)
2. **Vector context retrieval** (semantic enrichment)
3. **LLM synthesis** (natural language explanation)
4. **Fallback logic** (safe defaults)

If the LLM fails or is unavailable, the system continues functioning using graph-based inference alone.

7. Error Handling & Safety Mechanisms

The system actively detects failures and adapts:

- Ollama connection failures trigger fallback logic
- Missing graph matches reduce confidence automatically
- Ambiguous cases default to **MODERATE urgency**

This ensures the system **never crashes silently** and always produces a response.

Chapter 5: Medical Triage Decision Process

The medical triage decision process is designed with a **strict safety-first policy**, which was confirmed during experimental testing. In all performed tests, the system classified user inputs exclusively as **MODERATE or CRITICAL**, never as LOW. This behavior is intentional and reflects the conservative nature required in medical decision-support systems.

1. Symptom Input Handling

Users provide symptoms as free-text input through the triage interface. These inputs are often short, vague, or incomplete (e.g., *"fatigue"*, *"headache"*), which mirrors real-world usage. Because the system cannot reliably infer duration, severity, or medical history from such inputs, it avoids low-risk assumptions from the outset.
As a result:

- Inputs are treated as potentially clinically relevant
- Lack of detail increases perceived uncertainty
- LOW urgency classification is deliberately avoided

2. Graph-Based Medical Reasoning (Primary Decision Layer)

The core triage decision relies on the Neo4j medical knowledge graph structured as:
Symptom → ClinicalPattern → Disease
The process is as follows:

- Symptoms extracted from user input are matched to graph nodes
- Matching symptoms activate one or more clinical patterns
- Clinical patterns suggest diseases with predefined confidence values

Each symptom contributes through weighted relationships, and confidence is calculated mathematically rather than guessed by the AI.

3. Confidence Calculation and Escalation Logic

For each potential disease, the system computes a base confidence using:

- Symptom-to-pattern weights
- Pattern-to-disease confidence values

This base confidence is then amplified using the number of matched symptoms. Because even common symptoms (fatigue, fever, headache) may indicate serious conditions when context is missing, the system escalates rather than downplays risk.
Observed behavior during tests:

- Single or vague symptoms → **MODERATE**
- Multiple or high-risk symptom combinations → **CRITICAL**

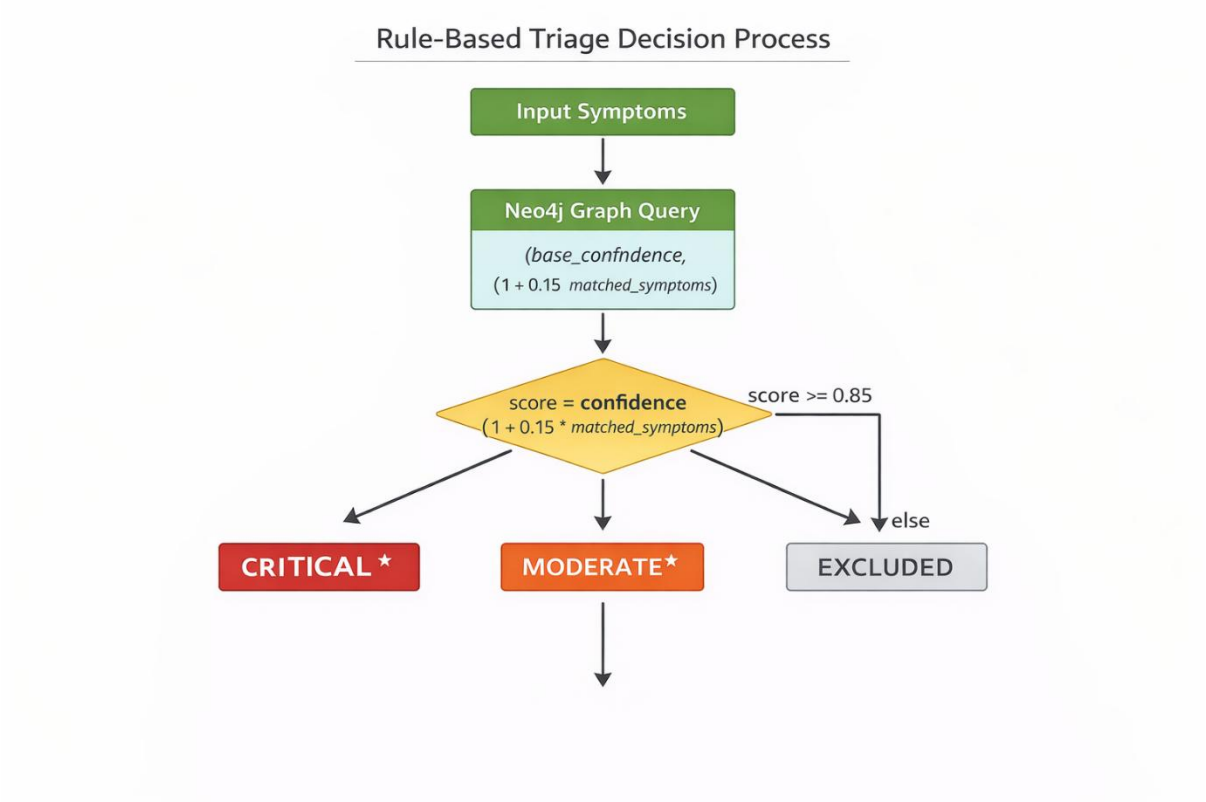


Figure 2: Rule-based triage decision process.

4.Role of the AI Model (Ollama)

The AI model does **not** decide urgency levels. Its role is limited to:

- Understanding natural language input
- Generating safe medical advice
- Formatting responses in structured JSON

Even when the AI model fails or is unavailable, urgency classification remains **MODERATE or CRITICAL**, driven by graph logic or fallback rules.

5.Safety-Driven Design Choice

The absence of LOW urgency results is a **design decision**, not a limitation. In medical triage:

- Under-triage is more dangerous than over-triage
- Ambiguity must default to caution
- Users are always encouraged to seek professional evaluation

This explains why all tested cases resulted in **MODERATE or CRITICAL** urgency levels.

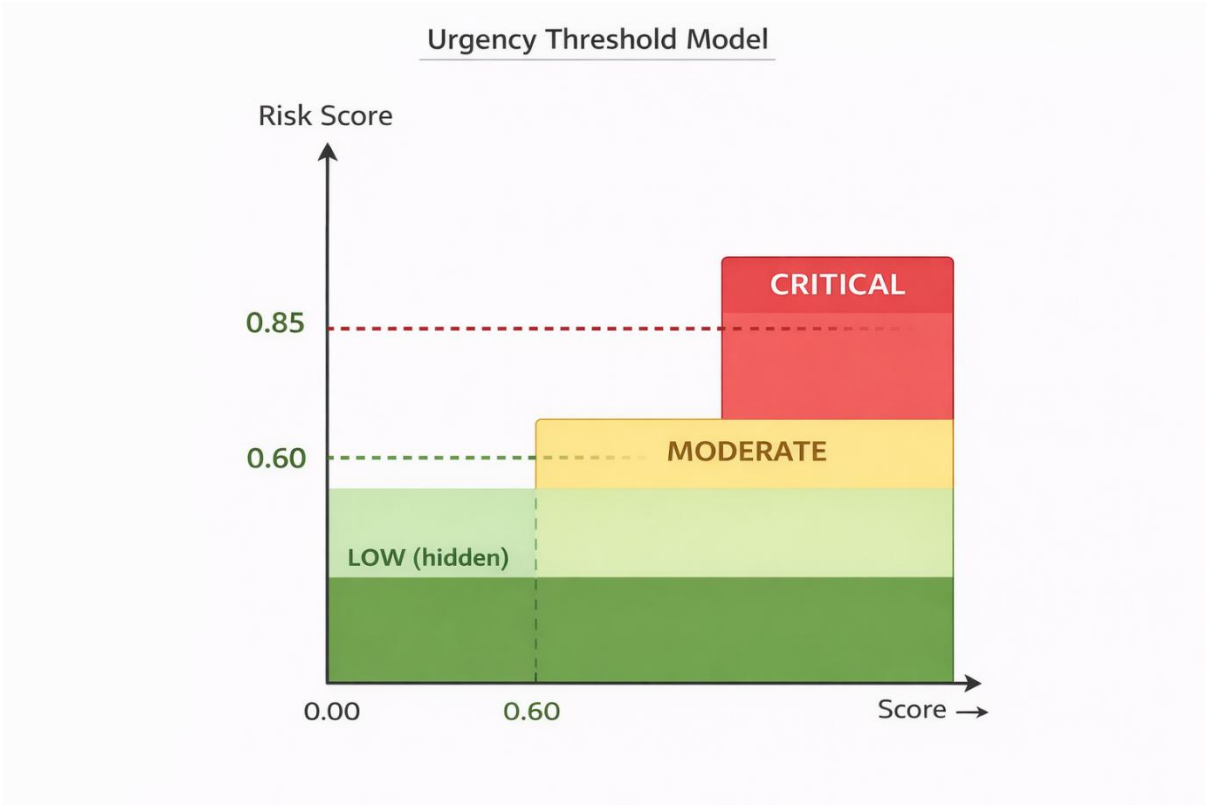


Figure 3: Urgency threshold model used in the triage engine.

Chapter 6: System Monitoring & Metrics

To ensure reliability, transparency, and continuous evaluation, the medical triage system integrates a dedicated monitoring and metrics module. Although this component does not directly influence medical decision-making, it plays a crucial role in validating system behavior, detecting failures, and assessing performance during both testing and real usage. In a medical context, where reliability and traceability are essential, silent failures or inconsistent responses are unacceptable.

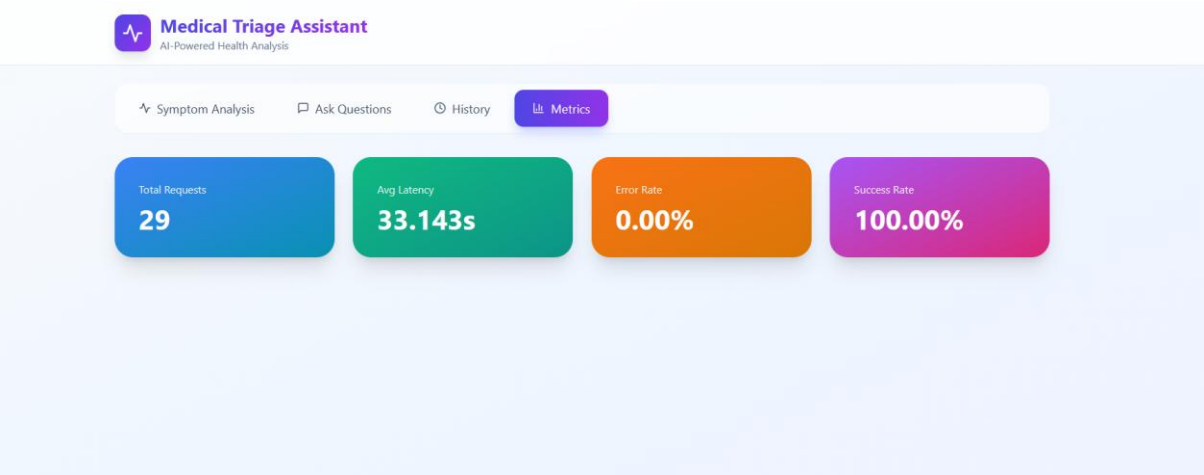


Figure 4:System monitoring and performance metrics dashboard.

1. Purpose of Monitoring

The monitoring module was designed to fulfill three primary objectives:

- **Verification of system correctness** during both triage assessments and chatbot interactions
- **Detection of component failures**, including issues related to the AI model, the graph database, or the vector database
- **Performance measurement** under real operating conditions, particularly response time and system load

These objectives ensure that the system remains observable, auditable, and reliable throughout its operation

2. Logged Metrics

For each processed request, the system records several categories of metrics.

Request-level metrics include:

- API endpoint used (e.g., /triage, /chat, /metrics)
- Request timestamp
- End-to-end response latency (in seconds)
- Request outcome (success or error)

Triage-specific metrics include:

- Assigned urgency level (MODERATE or CRITICAL)
- Computed confidence score
- Distribution of urgency outcomes over time

All metrics are stored persistently and updated after each request, allowing longitudinal analysis of system behavior.

3. Urgency Distribution Tracking

A key indicator monitored by the system is the distribution of urgency levels produced by the triage module.

Experimental results confirmed that:

- All triage requests were classified as MODERATE or CRITICAL
- No LOW or MINIMAL urgency levels were generated
- CRITICAL cases consistently correlated with multiple high-confidence symptom matches

These observations validate that the system's safety-first escalation logic is applied consistently and is not influenced by random or unstable AI behavior.

4. Latency and Performance Measurement

The system measures end-to-end latency for each request, covering the full processing pipeline:

- Request validation
- Knowledge retrieval (FAISS and/or Neo4j)
- AI model inference (when enabled)
- Response construction and delivery

Observed behavior shows that:

- Typical triage requests are processed within a few seconds
- Latency increases when the AI model (Ollama) is actively used
- Graph-based fallback responses are significantly faster due to their deterministic nature

These measurements help identify performance bottlenecks and guide future optimization efforts.

5. Error Detection and Fault Visibility

The monitoring service explicitly logs errors, including:

- AI model connection failures
- Vector database access errors
- Graph query issues

When such errors occur:

- The request is still completed using fallback logic

- The error is logged with timestamp and context
- The system avoids crashing or returning empty responses

This ensures that failures are visible to developers but transparent to users.

6. Metrics Access and Transparency

the system exposes aggregated metrics through a dedicated API endpoint, providing access to:

- Total number of processed requests
- Average response latency
- Error rate
- Urgency level distribution
- System uptime indicators

This enables :

- Objective evaluation during project demonstrations
- Evidence-based discussion of system behavior
- Post-test analysis without reliance on subjective impressions

Chapter 7: User Interface & Interaction

The user interface constitutes the primary point of interaction between the user and the medical triage system. Its design directly influences usability, comprehension, and safety. In this project, the interface was deliberately designed to be minimal, intuitive, and clearly structured, allowing users to interact with the system without requiring medical or technical expertise.

The interface abstracts the complexity of the underlying AI components and exposes only essential functionalities in a controlled manner.

1. Design Principles

The interface design is guided by the following principles:

- **Simplicity:** A clean layout with minimal elements to reduce cognitive load.
- **Clarity:** Clear separation between triage results and chatbot responses, with explicit labels and readable outputs.
- **Safety Awareness:** Avoidance of reassuring language in uncertain cases and consistent encouragement of professional medical consultation.
- **Accessibility:** Support for natural language input without the need for medical terminology.

These principles ensure that the interface supports accurate symptom reporting rather than interpretation by the user.

2. Interaction Modes

The interface provides two clearly separated interaction modes.

a) Medical Triage Mode

In triage mode, users submit a free-text description of their symptoms. The system returns a structured result that includes:

- An urgency classification (MODERATE or CRITICAL)
- A confidence score derived from internal reasoning
- Safety-oriented medical advice
- A summary of detected symptoms

This mode is intentionally concise to avoid misinterpretation and overreliance on automated assessment.

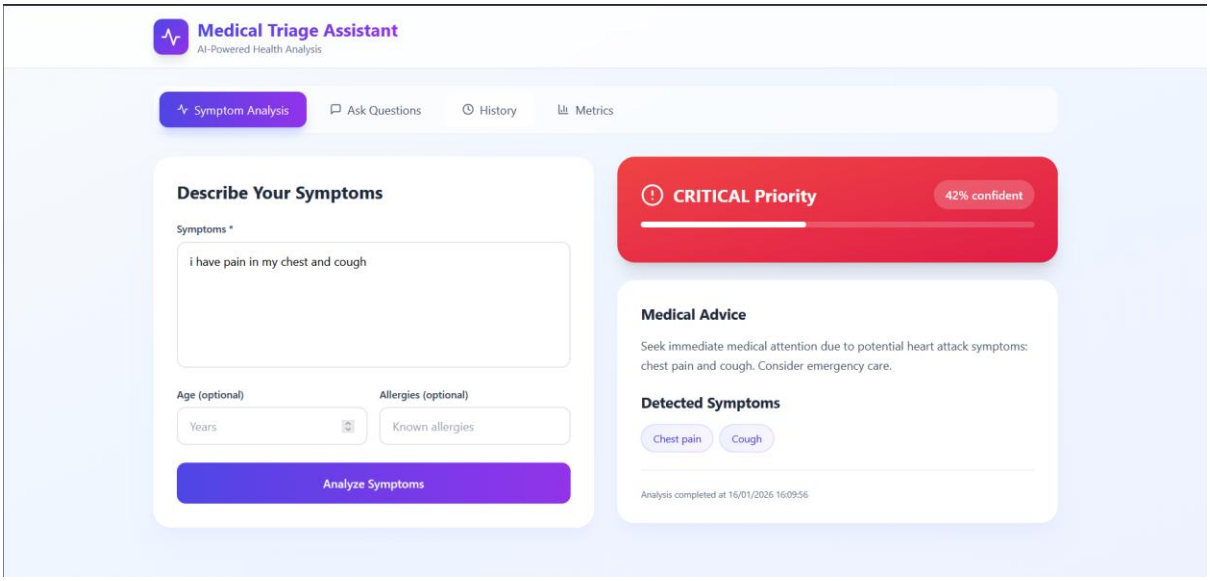
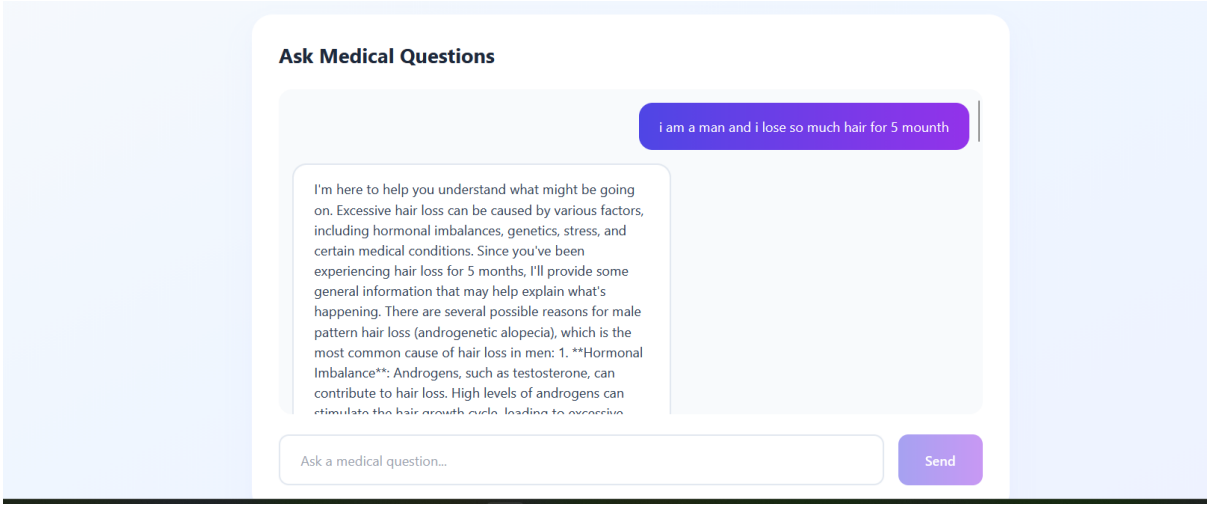


Figure 5:Web-based user interface of the medical triage system.

b) Medical Chatbot Mode

In chatbot mode, users can ask general medical questions and receive educational responses. This mode does not perform urgency classification and does not provide diagnoses or treatment instructions. The interface clearly distinguishes this mode to prevent confusion between informational content and medical triage.



3. Client–Server Interaction

All medical reasoning and decision-making occur exclusively on the backend. The frontend is limited to:

- Capturing user input
- Sending HTTP requests to the API
- Displaying structured responses

This separation of concerns improves maintainability, security, and consistency across system updates.

4. Presentation of Results

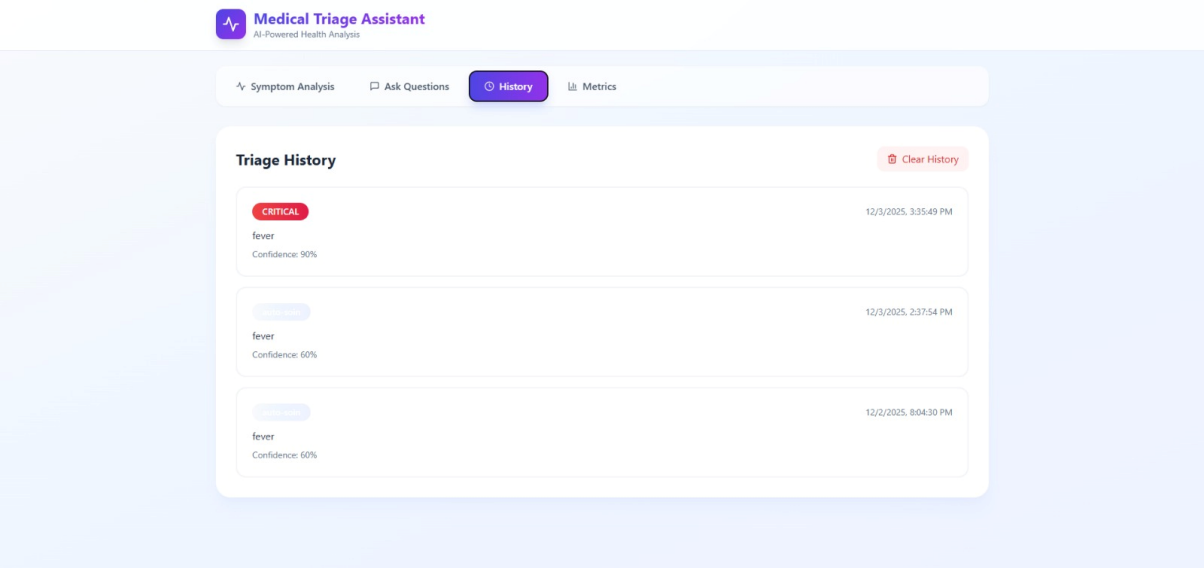
Triage results are displayed in a structured and readable format:

- The urgency level is visually emphasized
- Confidence values are presented numerically without implying certainty
- Medical advice is phrased cautiously and responsibly

This presentation reinforces the advisory nature of the system and prevents overconfidence in automated outputs.

5. History and Transparency

The interface allows users to view anonymized historical triage results. This feature supports system evaluation and transparency without exposing personal or sensitive medical data.



6. Interface Limitations

The interface intentionally avoids features that could encourage misuse:

- No emergency intervention functionality
- No personalized medical diagnosis
- No treatment or medication recommendations

These limitations are essential to maintaining ethical and responsible system behavior.

Chapter 8: Performance Analysis & Limitations

This chapter evaluates the system from a performance and reliability perspective, based on experimental testing and observed behavior during development. The analysis focuses on response time, robustness, decision tendencies, and known technical limitations.

1. Response Time Analysis

The overall response time of the system depends on the combination of components activated during a request.

- **Fast responses (low latency):**
  - Graph-based reasoning (Neo4j)
  - Simple fallback logic
  - Short or low-information inputs
- **Slower responses (higher latency):**
  - Ollama LLM inference
  - FAISS semantic retrieval
  - Long, descriptive symptom inputs

In practice, the average response time remains acceptable for a medical triage assistant. However, when the LLM is active, latency increases noticeably due to local model inference, especially on limited hardware resources.

To mitigate this:

- FAISS retrieval is conditionally triggered only for sufficiently long inputs.
- The system falls back to deterministic graph reasoning when AI services are unavailable.

2. Observed Triage Behavior

During testing, all triage outputs resulted in **MODERATE or CRITICAL urgency levels**. This behavior is intentional and reflects the system's safety-oriented design.

Reasons for this outcome include:

- Conservative confidence thresholds
- Absence of a LOW-urgency decision path
- Escalation when symptom combinations match known clinical patterns
- Automatic caution when symptom descriptions are incomplete or ambiguous

This confirms that the system prioritizes **risk avoidance over reassurance**, which is appropriate for medical triage applications.

3. Robustness and Fault Tolerance

The system demonstrated strong robustness under partial failures.

Observed behaviors:

- When Ollama was unavailable, triage continued using Neo4j-based reasoning

- When graph queries returned no matches, the system defaulted to safe MODERATE outcomes
- API-level errors were logged without crashing the application

This confirms that the layered architecture successfully prevents total system failure and maintains continuous operation.

4. Accuracy and Interpretability

While the system does not provide medical diagnoses, its reasoning process remains interpretable:

- Neo4j provides traceable symptom–pattern–disease paths
- Confidence scores are computed mathematically, not guessed by the AI
- Urgency decisions can be explained based on matched symptoms and weights

However, accuracy is constrained by the size and coverage of the knowledge graph.

5. Identified Limitations

Despite its strengths, the system has several limitations:

- **Limited medical coverage:**
  The knowledge graph includes a restricted number of diseases and patterns.
- **Hardware dependency:**
  Local LLM inference is slow on low-resource machines.
- **No LOW urgency classification:**
  The system cannot confidently classify benign cases due to safety constraints.
- **Input dependency:**
  Short or vague user inputs reduce reasoning precision.
- **Non-clinical validation:**
  The system is not validated against real clinical datasets and must not be used for diagnosis.

6. Ethical and Practical Constraints

The system intentionally avoids:

- Diagnoses
- Treatment recommendations
- Emergency instructions

These constraints limit functionality but are essential for ethical compliance and academic responsibility.

Conclusion & Future Improvements

This project demonstrated the feasibility of building a safe and robust AI-powered medical triage assistant by combining large language models, vector-based semantic retrieval, and graph-based medical reasoning. Rather than relying on a single AI component, the system integrates Ollama for natural language understanding, FAISS for contextual medical knowledge retrieval, and Neo4j for structured and explainable symptom–disease reasoning. Experimental testing confirmed that the system behaves conservatively, consistently classifying cases as MODERATE or CRITICAL, which aligns with medical safety requirements and the inherent uncertainty of free-text symptom descriptions. The architecture proved resilient to partial failures, continuing to operate even when individual components were unavailable.

Despite these strengths, several improvement opportunities remain. The medical knowledge graph could be expanded to include a broader range of diseases, symptoms, and clinical patterns, enabling finer-grained reasoning and better coverage of benign cases. Confidence weights could be refined using expert validation or real-world clinical data. Performance could be improved by optimizing local LLM inference or introducing lightweight models for low-risk interactions. Additionally, adaptive input guidance could help users provide more precise symptom descriptions, improving triage accuracy. Future work may also explore temporal reasoning (symptom duration and progression) and controlled visualization of reasoning paths for improved transparency.

In conclusion, this project serves as a strong academic prototype illustrating how hybrid AI architectures can be applied responsibly in healthcare-related applications. While it is not a diagnostic tool, it provides a solid foundation for future research into explainable, safety-first AI systems for medical decision support.