AI-Based Triage Navigator

Overview

The Streamlit Triage Navigator is a hybrid healthcare triage application that combines:

- A deterministic, rule-based medical triage engine (for safety and reliability), and
- An optional local Large Language Model (LLM) via Ollama (for better explanation quality and conversational interaction).

The system helps users understand:

- The urgency of their symptoms,
- What level of care they may need (Emergency, Urgent, Routine, or Home Care),
- And what their next steps should be,

while strictly following safety guardrails:

- No diagnosis
- No medications or prescriptions
- No replacement of professional medical advice

Key Features:

Safety First

- Always includes a medical disclaimer.
- Red-flag detection escalates cases to *Emergency* automatically.
- If LLM fails or becomes unavailable, the app falls back instantly to deterministic logic.

Robust Ollama Integration

- Persistent HTTP session with retries.
- Short request timeouts to avoid UI freezing.
- Cached health checks for fast UI startup.
- Automatic downgrade to fallback mode if Ollama fails.

Hybrid AI Logic:

Component:	Role
Rule Engine:	Final authority for urgency classification
LLM (Ollama):	Rewrites explanations & generates follow-up questions
Fallback Logic:	Guarantees function even without AI

System Architecture:

User

↓

Streamlit UI (Chat)

↓

Triage Wrapper

↓

Rule Engine (always runs)

↓

Optional LLM (Ollama)

↓

Safe Explanation + Next Steps

↓

User + JSON Output

Components Explained:

4.1 Ollama Configuration

OLLAMA_URL = "http://localhost:11434/api/generate"

OLLAMA_MODEL = "deepseek-r1:1.5b"

Defines where the local LLM is running and which model is used.

4.2 HTTP Session with Retries

@st.cache_resource

def get_http_session():

- Creates a persistent session.
- Adds automatic retries for network failures.
- Makes LLM calls faster and more stable.

4.3 Health Check System

check_ollama_available()

- Runs a quick ping at startup.
- Caches the result for 60 seconds.
- UI immediately shows whether LLM is active or fallback mode is being used.

4.4 Fallback Mechanisms

If LLM is unavailable:

- fallback_followup_question() generates safe questions.
- fallback_rewrite_explanation() produces simple, rule-based explanations.

This ensures zero downtime.

4.5 Medical Knowledge Base

KNOWLEDGE_BASE = [

chest pain,

fever,

headache

]

Each symptom contains:

- Emergency indicators

- Urgent indicators
- Home-care suggestions
- Weighted severity scores

This small dataset grounds the AI and prevents hallucination.

4.6 Triage Engine Logic

Red flags → Emergency

Else:

Emergency score ≥ 3 → Emergency

Urgent score ≥ 2 → Urgent

No scores → Home Care

Else → Routine

Clear, explainable, and hackathon-friendly.

4.7 Triage Wrapper

def triage(user_text, prefer_llm=True)

Workflow:

1. Detect red flags.
2. Score symptoms using rule engine.
3. Classify urgency.
4. Try LLM for explanation.
5. If LLM fails → fallback explanation.
6. Return structured result.

4.8 Streamlit UI

- Chat history stored in st.session_state.
- One-turn triage:

1. User gives symptoms.
2. AI asks one follow-up.
3. AI returns result + JSON.

Safety & Compliance

The app enforces:

- No medical diagnosis
- No drug names or dosages
- Always conservative escalation
- Clear disclaimers in every result

This is critical for healthcare demos and judges.

How to Run

pip install streamlit requests urllib3

ollama pull deepseek-r1:1.5b

ollama serve

streamlit run app.py