System Design Document

AI Health Assistant App

the role of Generative AI on Tigray Healthcare

Dina Ibrahim NIGUSSE

# Introduction

This document presents the software architecture and system design for the **AI Health Assistant**, a mobile-first, offline-capable healthcare chatbot. It outlines the major design decisions, architecture, system components, and interactions. The system is designed to provide accessible first-aid guidance, symptom education, and mental health support, particularly for communities in Tigray with limited access to healthcare. This document builds upon the functional and non-functional requirements specified in the Requirements Analysis Document (RAD), providing a solid technical foundation for system implementation.

## Purpose of the System

The AI Health Assistant aims to decrease barriers to healthcare information as a multilingual, easily accessible, offline-first application. This application is designed for individuals who are situated in rural or conflict-affected areas with limited healthcare services and inconsistent internet access.

* For individuals, the app provides access to first-aid instructions, symptom awareness, mental health resources, and referrals.
* For healthcare workers, the app will serve as a community resource for patient education and support.
* For the larger community, it provides timely self-care, support for trauma recovery, and health awareness in Tigrinya, Amharic, and in English.

Users will be assured safety through reliance on a rule-based knowledge base (based on WHO and Ethiopian Ministry of Health guidelines) to provide options on symptoms, first-aid, and mental health challenges. It will also provide a generative AI solution (when internet access is available) to supplement the response. This application will act as a reliable, private, and scalable resource to support healthcare provision and delivery in underserved areas.

## Design Goals

* Usability:
  + Users should be able to access the chatbot
  + The chat interface must be simple, mobile-first and multilingual
  + Users with limited digital literacy should be able to interact with the system with minimal training
  + The UI should clearly indicate whether the response is rule-base, or AI generated
* Utility
  + Provide **first-aid instructions** and **health education** based on trusted sources (WHO, Ethiopian MoH).
  + Offer **mental health support resources** in a safe and stigma-free environment.
  + Enable **offline usage** for rule-based responses to accommodate areas with poor internet.
  + Provide optional **AI fallback** for queries not covered by the rule engine.
  + Allow session tracking and logging to support quality improvement and outbreak monitoring (with privacy safeguards).
* Performance
  + Rule-based responses should be retrieved and displayed in **< 2 seconds**.
  + AI fallback responses should be processed in **< 10 seconds** (local model or cloud, depending on availability).
  + The system should scale to handle **multiple concurrent users** without significant slowdown.
  + Response caching should be used to improve performance for frequently asked questions.
* Dependability
  + Ensure **data privacy and security**: no personal identifiers stored by default, logs anonymized.
  + Use **secure authentication** for clinician/admin access to update rules and review flagged responses.
  + Provide **error handling and fallback messages** when the system cannot process a query.
  + Ensure offline availability of the rule engine to avoid dependency on unreliable internet connections.
  + Include **automatic data backup** of knowledge base and logs where internet or local storage permits.
  + Ensure accurate translation between languages using Google API when there is internet access, and a fallback to rule-based multilingual JSON KB when offline.

# System Architecture

The AI Health Assistant is a hybrid, offline-first chatbot with a modular architecture. It consists of a presentation layer (frontend), an application layer (backend), a rule-based knowledge base, an optional AI module (generative fallback), and a data layer (storage and logs).

The overall philosophy is rule-first, AI-fallback:

* For health guidance core (first-aid, trauma recovery, health education), these queries are processed by deterministic rule-based engine to offer reliability and offline capability.
* For queries not covered by rules, the generative AI model (local lightweight LLM or cloud-based model if available) is leveraged.
* The question and replies will pass through safety and filtering layer to ensure safety of replies and disclaimers.

## High-level Architecture Layers

1. Frontend

* Built as a **mobile-first Progressive Web App (PWA)** using Vue.js.
* Provides a simple chat interface with multilingual input/output (Tigrinya, Amharic, English).
* Supports offline caching for knowledge base lookups.

1. Backend

* Implemented using **FastAPI** for lightweight, asynchronous API handling.
* Routes user queries through the **rule-based engine** first.
* Falls back to the **AI Module** if no rule match is found.
* Manages sessions, logging, and communication with the database.

1. Translation Module

* A translation component between frontend and backend
* Frontend sends text in any supported language.
* Translation Module detects language, translates into English if needed.
* Rule Engine / Knowledge Base or AI processes English text.
* Response is translated back into the original language.
* Ensures normalization of low-resource languages (Tigrigna) into English for consistent intent matching.

1. Knowledge Base (Rule-based Engine)

* Stores curated health guidance (WHO, Ethiopian MoH, trauma protocols) in JSON/SQLite.
* Provides **fast, offline responses** for common symptoms, first-aid, and mental health support.

1. AI Module (Generative Fallback)

* Optional component using **local LLMs** (e.g., GPT4All, LLaMA distillations) or **cloud-hosted models**.
* Handles conversational queries not covered by rules.
* Responses pass through a **safety filter** before being returned to users.
* Used only when internet is available and the query cannot be resolved by the rule engine.

1. Safety & Filtering Layer

* Detects red-flag symptoms (e.g., chest pain, severe bleeding).
* Blocks unsafe or hallucinated AI outputs.
* Ensures all responses include disclaimers when appropriate.

1. Data Layer (Storage and Logging)

* Uses **SQLite** for lightweight session storage, chat logs, and feedback.
* Stores anonymized interaction data for quality improvement and outbreak monitoring.
* Supports privacy by default — no personal identifiers are stored.

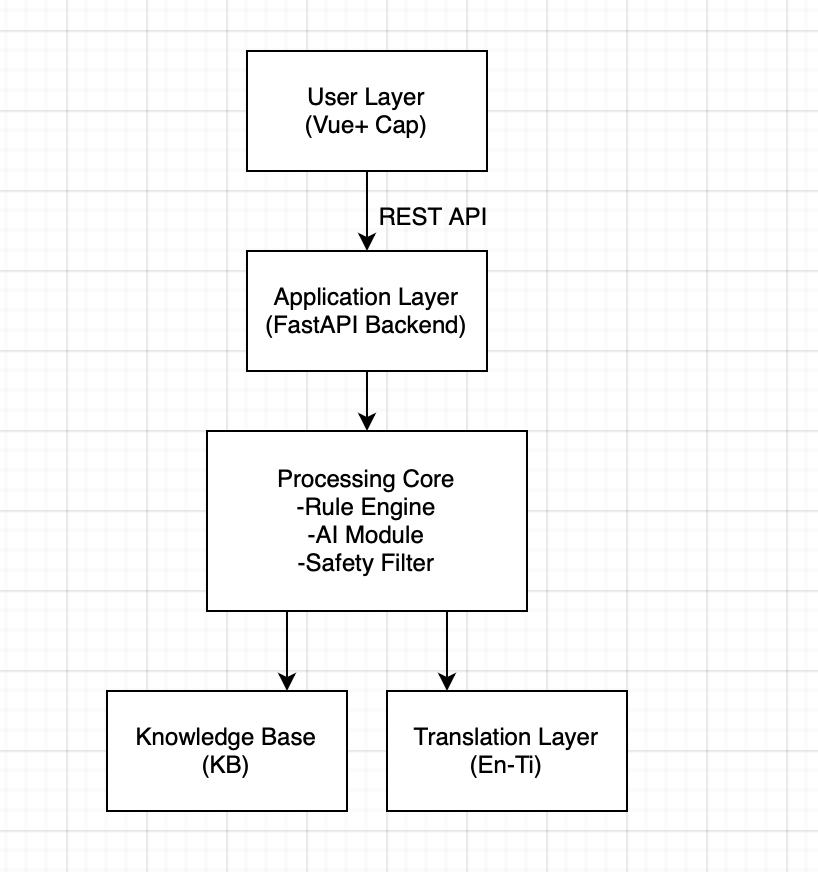


Fig 1. System Architecture Diagram

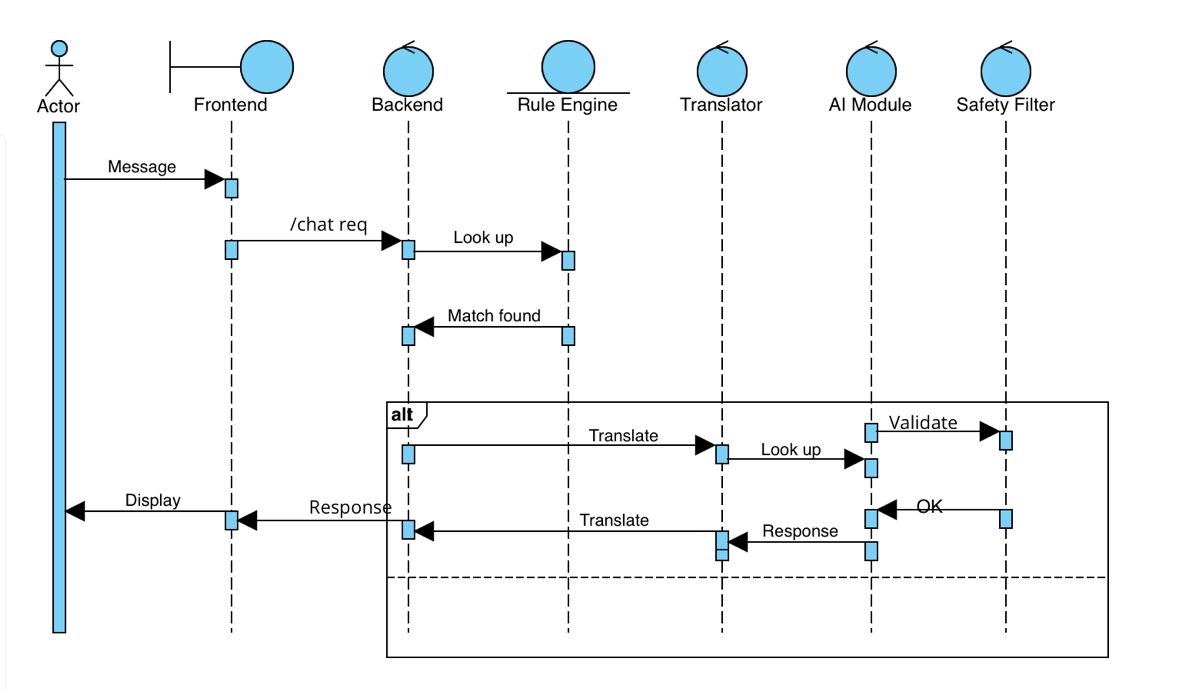


Fig 2. Sequence Diagram

## Data Design – Knowledge Based Schema

The **Knowledge Base (KB)** stores health guidance rules in JSON format. Each entry contains:

* **Intent/Condition Name** → unique identifier (e.g., bleeding, severe\_bleeding).
* **Keywords** → list of words/phrases used for rule matching.
* **Steps** → step-by-step guidance to show the user.
* **Follow ups** → optional decision questions for further triage.
* **Escalation** → message advising when to seek emergency care.

Example Entry:

"bleeding": {

"keywords": ["bleeding", "cut", "wound", "blood"],

"steps": [

"Wash your hands if possible before touching the wound.",

"Apply firm pressure with a clean cloth or bandage to stop the bleeding.",

"Keep the injured part elevated above the heart if possible."

],

"followups": [

{

"question": "Is the bleeding spurting or unusually heavy? (yes/no)",

"yes\_intent": "severe\_bleeding",

"no\_intent": null

}

],

"escalation": "If bleeding does not stop after 10 minutes or is severe, seek emergency medical care."

},

"severe\_bleeding": {

"keywords": ["severe bleeding", "spurting blood", "arterial"],

"steps": [

"Call for emergency help immediately if available.",

"Apply very firm pressure and pack the wound with clean cloth.",

"If trained and a tourniquet is available, apply it above the wound.",

"Keep the person calm and monitor breathing."

],

"escalation": "Seek urgent hospital care immediately."

}

## Deployment and System Environment

* Frontend Deployment: Delivered as a Progressive Web App (PWA), packaged into a native Android app using Capacitor
* Mobile Platform: Target environment is Android OS (tested on emulator)
* Backend Deployment: FastAPI backend runs locally on mobile devices (for offline KB) and connects to online AI/translation services when internet is available
* Distribution: Application can be sideloaded on Android or deployed via Google Play.