**Software Requirements Specification (SRS)**

**Project: Doc-in-the-Box - Phase 1+ (Voice-Enabled Medical Triage PoC)**

**Version:** 1.0  
**Authors:** Mayank, Prince, Upendra  
**Status:** Draft

**1. Introduction**

**1.1 Purpose**

This document describes the requirements for a Voice-Enabled Proof of Concept (PoC) web application for a medical triage assistant. The system will allow users to input symptoms via voice or text and receive a probabilistic, cited differential diagnosis with an urgency level. The primary goal is to demonstrate the feasibility of the core AI technology with a high-impact, accessible user interface.

**1.2 Scope**

This PoC will be a web application with a limited knowledge base focusing on a specific set of infectious diseases (e.g., Malaria, Typhoid, Viral Fever). It is strictly a triage assistant and will **not** provide definitive diagnoses. The final output will always include a disclaimer to consult a real doctor. Voice output (Text-to-Speech) is designated as a stretch goal.

**1.3 Definitions, Acronyms, and Abbreviations**

* **PoC:** Proof of Concept
* **UI:** User Interface
* **API:** Application Programming Interface
* **LLM:** Large Language Model
* **RAG:** Retrieval-Augmented Generation
* **STT:** Speech-to-Text
* **TTS:** Text-to-Speech
* **API:** Application Programming Interface

**2. Overall Description**

**2.1 Product Perspective**

This is a standalone web application. It will consist of:

* A React.js frontend served by a development server.
* A Python FastAPI backend server.
* Connections to external APIs: LLM (OpenAI/Anthropic) and STT (OpenAI Whisper). TTS (Google/Amazon) will be added as a stretch goal.

**2.2 User Classes and Characteristics**

* **Primary User:** A demonstration user (e.g., professor, investor) with access to a modern web browser.
* **End-User Persona (Future Consideration):** An adult in a rural area with potentially low literacy, speaking a vernacular language (e.g., Hindi, Tamil), with limited access to immediate medical care.

**2.3 Operating Environment**

* **Frontend:** Modern browsers with JavaScript enabled (Chrome, Firefox, Safari).
* **Backend:** Python 3.10+ environment.
* **Deployment:** Localhost for the PoC demonstration. Can be containerized with Docker for portability.

**2.4 Design and Implementation Constraints**

* Must process user input in vernacular languages (Hindi, Tamil).
* Must generate a diagnosis that is probabilistic, cited, and includes an urgency indicator.
* Must include a clear medical disclaimer on the UI and in any audio output.
* The AI must never state a definitive diagnosis.

**3. System Features**

**3.1 Feature 1: Multi-Modal Symptom Input**

* **Description:** The user shall be able to describe their symptoms either by typing into a text field or by speaking into their microphone.
* **Sub-Feature 1.1: Voice Input (STT)**
  + The system shall capture audio via the browser's microphone API.
  + The system shall convert the spoken vernacular language (Hindi/Tamil) to text using the OpenAI Whisper API.
* **Sub-Feature 1.2: Text Input**
  + The system shall provide a text input field as a fallback.

**3.2 Feature 2: AI-Powered Triage Engine (RAG)**

* **Description:** The system shall analyze the symptom text and generate a differential diagnosis.
* **Sub-Feature 2.1: Knowledge Retrieval**
  + The system shall query a vector database (ChromaDB) containing chunks of text from trusted medical manuals (e.g., WHO guidelines, MSF manuals).
* **Sub-Feature 2.2: Diagnosis Synthesis**
  + The system shall use an LLM (GPT-4 Turbo or Claude 3) with a carefully engineered prompt to synthesize the retrieved context into a structured diagnosis.
  + The diagnosis must include:
    1. **Urgency Level:** One of High, Medium, or Low.
    2. **Possible Conditions:** A list of 2-3 conditions, each with:
       - A percentage likelihood.
       - A one-sentence rationale citing the retrieved context.
    3. **A clear disclaimer.**

**3.3 Feature 3: Diagnosis Presentation (UI)**

* **Description:** The system shall display the diagnosis results in a clear, user-friendly interface.
* **Requirements:**
  + The urgency level shall be displayed as a prominent, color-coded badge (e.g., Red for High).
  + The list of possible conditions shall be easy to read.
  + The medical disclaimer shall be displayed in bold, unmistakable text.

**3.4 Feature 4: Voice Output - TTS (Stretch Goal)**

* **Description:** The user shall be able to hear the diagnosis read aloud in the target vernacular language.
* **Requirements:**
  + A "Play Audio" button shall be present next to the diagnosis results.
  + Clicking the button shall play the generated audio of the diagnosis being read.
  + The audio shall be generated using a cloud TTS API (Google Cloud TTS or Amazon Polly).

**4. External Interface Requirements**

**4.1 User Interfaces**

* The UI shall be a single-page application (SPA) with the following components:
  + A header with the project name.
  + A large, prominent microphone button for voice input.
  + A text input field with a submit button.
  + A results section to display the urgency, conditions, and disclaimer.
  + A "Play Audio" button (stretch goal).

**4.2 Hardware Interfaces**

* Requires a microphone for voice input.
* Requires speakers/headphones for audio output (stretch goal).

**4.3 Software Interfaces**

* **OpenAI Whisper API:** For Speech-to-Text conversion.
* **OpenAI GPT-4 API or Anthropic Claude API:** For the LLM inference.
* **Google Cloud TTS API or Amazon Polly API (Stretch Goal):** For Text-to-Speech synthesis.

**4.4 Communications Interfaces**

* Frontend and Backend will communicate via HTTP RESTful API calls using JSON.

**5. Non-Functional Requirements**

* **Performance:** The system should return a diagnosis within 10-15 seconds of receiving input.
* **Reliability:** The LLM prompt must be engineered to be highly robust and consistently return the required JSON format.
* **Usability:** The interface must be simple and intuitive, requiring minimal instruction for a demo.
* **Security:** API keys for external services must be stored securely using environment variables and not hardcoded.

**Technology Stack & Implementation Plan**

**Methodology: Phased Agile Development**

The project will be executed in two clear phases to manage scope and risk effectively.

* **Phase 1 (Core Product):** Implement Features 1, 2, and 3 (Voice Input, AI Triage, UI Display). **This is the mandatory minimum viable product (MVP).**
* **Phase 2 (Stretch Goal):** Implement Feature 4 (Voice Output) **only after Phase 1 is fully complete and demo-ready.**

**Technology & Libraries**

| Component | Technology | Libraries / APIs | Purpose |
| --- | --- | --- | --- |
| **Frontend** | React.js 18 | axios, react-microphone, MUI or Chakra UI | User Interface, audio capture, API calls |
| **Backend** | FastAPI (Python) | uvicorn, python-multipart, httpx, langchain | Main server, orchestration, handle file uploads |
| **AI & Data** | - | openai (Whisper & GPT), chromadb | STT, LLM, Vector Database for RAG |
| **Voice Output** | - | google-cloud-texttospeech or boto3 (for Polly) | TTS Synthesis (Stretch Goal) |

**Work Distribution & Responsibilities**

| Team Member | Primary Role | Key Responsibilities |
| --- | --- | --- |
| **Mayank** | **Backend & AI Lead** | **1. RAG Pipeline:** Setup ChromaDB, ingest medical data, engineer the core LLM prompt. **2. Backend Core:** Build the main FastAPI server and /diagnose endpoint. **3. Integration:** Connect the backend to the LLM API. |
| **Prince** | **Frontend & Voice Input Lead** | **1. UI/UX:** Build the entire React.js application (buttons, input fields, results display). **2. Voice Input:** Implement microphone capture and integrate with the Whisper API. **3. Core Integration:** Connect frontend to the backend's /diagnose endpoint. |
| **Upendra** | **Voice Output & Integration Lead** | **1. TTS Research:** Research and select a TTS API (Google/Amazon), setup accounts. **2. Backend for TTS:** Create a new FastAPI endpoint (e.g., /synthesize\_speech) to generate audio. **3. Frontend for TTS:** Work with Prince to add the "Play Audio" button and connect it to the new endpoint. **4. Phase 1 Support:** Assist with testing and bug fixes for the core product. |



Mayank - Backend & AI Lead

Prince - Frontend & Voice Input Lead

Upendra - Voice Output & Integration Lead