# Project Planning Document: Specialized English-only Medical Advisor (Few Symptoms)

## Overview

This project aims to develop a conversational AI assistant that accepts symptom descriptions in English and provides triage-level recommendations (Monitor at Home, See a Doctor, or Emergency). It uses GPT-4 for reasoning and follow-up questioning, Whisper for optional speech-to-text input, and a structured decision-making pipeline grounded in public triage guidelines such as those from Mayo Clinic or the CDC. The system will be delivered as a web-based interface using Streamlit or Gradio.

## Week 0 – Preparation & Proof of Concepts (PoCs)

Objectives:

* • Validate API keys and access
* • Install tools and dependencies
* • Run initial PoCs to de-risk speech input and LLM integration

### PoC 1 – Symptom Input to Triage Suggestion (GPT-4)

Input: Text symptom such as 'I have a sore throat and fever.'

Expected Output: A triage response such as 'You may monitor at home unless you develop a high fever or difficulty swallowing.'

Success Criteria: GPT-4 provides a coherent and clinically relevant response based on symptoms.

### PoC 2 – Speech-to-Text Input using Whisper

Input: Audio file of spoken symptoms (e.g., .wav format).

Expected Output: Accurate transcription of symptom description.

Success Criteria: Whisper accurately converts English audio into text (≥90% accuracy).

### PoC 3 – Decision Flow Using Prompt Engineering

Input: Initial symptom + 1–2 follow-up questions scripted through prompt chaining.

Expected Output: Follow-up questions and final triage recommendation with rationale.

Success Criteria: Chain-of-thought prompts produce logical step-by-step reasoning.

## Tools and Libraries

• Python 3.10+

• openai (for GPT-4 and Whisper APIs)

• langchain (for optional agent-based workflows)

• streamlit or gradio (for frontend interface)

• graphviz or networkx (for visualizing reasoning path)

• pandas, requests, tiktoken

## Week 1 – Core System Implementation

• Define scope: 5–7 symptoms (fever, sore throat, cough, chest pain, headache, fatigue, rash)

• Finalize triage criteria based on publicly available medical guidelines

• Create prompt templates for GPT-4 to generate clarifying questions and triage

• Build the frontend using Streamlit or Gradio

• Test and evaluate early prototypes using manually created test cases

## Week 2 – Speech Input and Explainability

• Integrate Whisper for speech-to-text conversion

• Add disclaimers for safety and scope

• Implement visualization of triage logic using decision trees or chain-of-thought flowcharts

• Add optional education step for self-care recommendations

## Week 3 – Evaluation and Final Presentation Prep

• User testing with simulated symptom cases

• Evaluate triage accuracy and clarity of recommendations

• Finalize UI and perform error handling

• Prepare report, video, and presentation slides

## Evaluation Plan

• Accuracy of Triage: Compare output with gold-standard medical guidelines for 10–15 test scenarios

• Relevance of Follow-up Questions: Reviewed for logical progression and medical value

• Clarity of Output: Evaluated for readability and helpfulness by a small peer review group

## Expected Deliverables

• Fully functional web-based advisor prototype

• 5-page ACM-style report documenting problem, methods, results, and evaluation

• 5-minute demo video showing example symptom interactions and reasoning path

• Slide deck for class presentation

Here's a clear, **detailed step-by-step guide** to help you get started and confidently build your **Specialized English-only Medical Advisor** project.

## ✅ ****Phase 0: Setup & Kickoff (Day 1–2)****

### 1. ****Set Up Your Development Environment****

* Install Python 3.10+ (via [python.org](https://www.python.org/) or Anaconda)
* Create a virtual environment:
* python -m venv advisor\_env
* source advisor\_env/bin/activate # or .\advisor\_env\Scripts\activate on Windows
* Install essential libraries:
* pip install openai streamlit gradio langchain whisper tiktoken pandas graphviz

### 2. ****Get API Keys****

* **OpenAI API**: [Sign up](https://platform.openai.com/) and get your key from your account page.
* Save your key as an environment variable:
* export OPENAI\_API\_KEY=your-key # or setx on Windows

### 3. ****Create a GitHub Repository****

* Initialize a GitHub repo to track progress and share your final submission.
* Suggested structure:
* /data
* /notebooks
* /app
* /outputs
* README.md

## ✅ ****Phase 1: Proof of Concepts (Day 3–5)****

### 4. ****PoC 1 – GPT-4 Triage Output from Symptom****

* Create a script triage\_gpt\_test.py
* Test prompt: “I have a sore throat and headache.”
* Confirm GPT-4 returns coherent triage suggestions.

### 5. ****PoC 2 – Whisper Audio to Text****

* Record a simple WAV audio file saying: “I have chest pain and shortness of breath.”
* Use:
* whisper your\_audio.wav --model base
* Output should be >90% accurate transcription.

### 6. ****PoC 3 – Clarifying Question + Decision Flow****

* Write chain-of-thought prompt like:
* Given these symptoms: {input}
* Ask one follow-up question to determine urgency.
* Then suggest one of:
* - Monitor at home
* - See a doctor
* - Go to the emergency room
* Try a few combinations in a script.

## ✅ ****Phase 2: Core App Development (Day 6–10)****

### 7. ****Define the Symptom Scope****

* Choose 5–7 common symptoms:
  + Fever
  + Sore throat
  + Cough
  + Rash
  + Chest pain
  + Headache
  + Fatigue

### 8. ****Gather Reference Guidelines****

* Manually collect triage recommendations from:
  + [Mayo Clinic](https://www.mayoclinic.org/)
  + [MedlinePlus](https://medlineplus.gov/symptoms.html)
* Summarize symptom → triage mappings into a JSON or CSV file.

### 9. ****Design Prompt Templates****

* One for clarification:
* A patient says they have: {symptoms}.
* Ask 1 follow-up question to help assess urgency.
* One for triage decision:
* Based on: {symptoms + answer}
* What level of urgency does this require?
* Choose from:
* - Emergency
* - See a doctor
* - Monitor at home
* Provide reasoning.

### 10. ****Build the App Interface****

* Use **Streamlit** or **Gradio** to:
  + Accept symptom input (text box)
  + Optionally record/upload audio (optional)
  + Display GPT response
  + Show follow-up question and final recommendation

## ✅ ****Phase 3: Explainability & Enhancements (Day 11–14)****

### 11. ****Add Explanation of Reasoning****

* Include reasoning text from GPT output
* Add a **simple diagram** of decision flow using:
* import graphviz

### 12. ****Add Whisper Integration into App (Optional)****

* Capture audio using microphone or upload.
* Transcribe with Whisper.
* Pipe result into the GPT pipeline.

### 13. ****Add Safety & Disclaimers****

* Add clear text:

“This tool does not provide medical advice. It is for educational purposes only.”

## ✅ ****Phase 4: Testing, Evaluation & Presentation (Day 15–21)****

### 14. ****Build a Test Set of Symptom Scenarios****

* Example:
  + “I have a fever and sore throat.” → “See a doctor”
  + “I have chest pain and shortness of breath.” → “Emergency”
* Use ~10–15 cases.

### 15. ****Create an Evaluation Rubric****

| **Metric** | **Criteria** |
| --- | --- |
| Triage accuracy | Does it match medical guidelines? |
| Clarifying question value | Was it medically relevant? |
| Explanation clarity | Is the logic clear to users? |
| UI simplicity | Can a non-technical person use it? |

### 16. ****Record Video Demo****

* Use screen recording to show:
  + UI flow
  + Several user examples (text + audio)
  + Reasoning path visualization

### 17. ****Write Report (ACM Format)****

Sections to include:

* Abstract
* Introduction
* Background and Motivation
* Methodology
* Implementation
* Evaluation
* Results
* Discussion
* Limitations
* Conclusion
* References

### 18. ****Prepare Presentation Slides****

* Keep it to 10–15 slides
* Use visuals from UI and results
* Include a clear narrative: problem → solution → impact

