Based on the **Medical Advisor Project Plan** document, here's a **detailed and final plan** that encompasses **all phases** and **PoCs** (Proof of Concepts) to address **technical risks** and ensure smooth project execution.

**Complete Project Plan for Specialized English-only Medical Advisor**

**Phase 0: Preparation & Proof of Concepts (PoCs)**

**Duration**: **Week 0 (Day 1–2)**  
**Objective**: Validate APIs, set up environment, and run initial PoCs to de-risk core functionalities.

1. **Set Up Development Environment**:
   * Install Python 3.10+ (via python.org or Anaconda).
   * Create 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 whisper streamlit gradio langchain python-dotenv pandas graphviz tiktoken
2. **API Key Setup**:
   * Obtain OpenAI API key.
   * Store the key in a .env file:
   * OPENAI\_API\_KEY=your-key-here
   * Add .env to .gitignore:
   * echo ".env" >> .gitignore
3. **PoC 1: Symptom Input to Triage Suggestion**:
   * Create a script to test GPT-4 for generating triage suggestions from symptoms.
   * input = "I have a sore throat and fever."
   * output = GPT-4 triage suggestion
   * **Expected Output**: GPT-4 should provide a coherent triage recommendation.
4. **PoC 2: Whisper Speech-to-Text Integration**:
   * Test Whisper’s ability to transcribe audio files into text.
   * whisper symptom\_audio.wav --model base
   * **Expected Output**: The transcription accuracy should be ≥90%.
5. **PoC 3: Clarifying Question and Decision Flow**:
   * Chain prompts to simulate GPT-4 asking follow-up questions based on symptoms.
   * "Given these symptoms: {input}, ask a follow-up question."
   * **Expected Output**: GPT-4 should ask relevant follow-up questions and provide triage recommendations.

**Phase 1: Core System Implementation**

**Duration**: **Week 1 (Day 3–5)**  
**Objective**: Build the core system that processes symptom input, generates clarifying questions, and makes triage decisions.

1. **Define Symptom Scope**:
   * Choose 5-7 symptoms to focus on, such as **fever, sore throat, cough, chest pain, headache**.
2. **Gather Reference Guidelines**:
   * Collect publicly available medical triage guidelines (e.g., Mayo Clinic, MedlinePlus).
3. **Design GPT-4 Prompt Templates**:
   * Develop prompt templates for:
     + **Clarification**: Asking follow-up questions.
     + **Triage Decision**: Determining urgency levels (Emergency, Doctor Visit, Monitor at Home).
4. **Build Frontend Interface**:
   * Use **Streamlit** or **Gradio** to create an interactive UI where users can:
     + Input symptoms.
     + Receive triage suggestions and follow-up questions.
5. **Test Prototype**:
   * Manually test with test cases to ensure proper functionality.

**Phase 2: Speech Input and Explainability**

**Duration**: **Week 2 (Day 6–10)**  
**Objective**: Add speech-to-text capabilities and improve explainability of decision-making.

1. **Integrate Whisper for Speech Input**:
   * Use **Whisper** to capture and transcribe spoken symptoms.
   * Integrate this with the GPT-4 decision-making pipeline.
2. **Decision Flow Visualization**:
   * Use **Graphviz** or **NetworkX** to visualize decision paths from symptom input to triage recommendation.
   * Add reasoning steps to clarify how triage decisions were made.
3. **Safety and Disclaimers**:
   * Add clear disclaimers:  
     **“This tool does not provide medical advice. For educational purposes only.”**

**Phase 3: Testing, Evaluation, and Presentation**

**Duration**: **Week 3 (Day 11–15)**  
**Objective**: Conduct tests, collect feedback, finalize the prototype, and prepare for presentation.

1. **Build a Test Set of Symptom Scenarios**:
   * Create 10-15 test cases based on real-life symptoms:
     + Example: "I have chest pain and shortness of breath." → **Emergency**.
2. **Evaluation Rubric**:
   * Assess:
     + **Triage Accuracy**: Compare against gold-standard guidelines.
     + **Clarifying Question Quality**: Ensure medically relevant questions.
     + **Clarity of Output**: Ensure clarity in user-facing responses.
     + **UI Usability**: Test UI with non-technical users.
3. **User Testing**:
   * Simulate user input and evaluate the system's responses.
4. **Record Demo Video**:
   * Record a 5-minute demo showcasing:
     + Symptom input (both text and audio).
     + Decision-making flow and triage recommendations.
5. **Prepare Report and Slides**:
   * Write a **5-page ACM-style report** covering:
     + **Introduction**, **Methodology**, **Implementation**, **Evaluation**.
   * Create a **slide deck** for the class presentation.

**Phase 4: Final Deployment and Maintenance**

**Duration**: **Week 4 (Day 16–21)**  
**Objective**: Finalize deployment, ensure stability, and prepare for real-world usage.

1. **Deploy the System**:
   * Deploy the system to a **cloud platform** (AWS, Google Cloud).
   * Ensure the system is accessible for users online.
2. **Final Testing and Bug Fixes**:
   * Test the deployed system with live users.
   * Fix any bugs or issues that arise.
3. **Monitor System Performance**:
   * Ensure that the system can handle concurrent users and large traffic.
4. **Prepare for Long-Term Maintenance**:
   * Plan for future updates to support additional symptoms and improvements.

**Final Deliverables:**

* **Web-based Advisor Prototype**: Fully functional system deployed online.
* **5-page ACM-style Report**: Documenting the problem, methods, results, and evaluation.
* **5-Minute Demo Video**: Showcasing user interactions and the decision-making flow.
* **Presentation Slides**: For final class presentation.

**Final Thoughts:**

* **PoCs in Phase 0** are crucial to de-risk the core functionalities (GPT-4, Whisper integration, decision flow).
* **Phase 1** ensures a reliable core pipeline.
* **Phase 2 and 3** focus on enhancing the user interface, explainability, and ensuring the system is medically valid.

If you have specific preferences for tasks in each phase or need additional guidance on implementation, feel free to let me know!

Based on the previous work, you are currently **in Phase 1** of the project, and specifically, you have completed **Proof of Concept (PoC) 1 and PoC 2**. Here’s a recap of what’s done and where we are in the project:

**Current Progress:**

1. **Phase 0: Proof of Concept (PoCs)**:
   * **PoC 1 (Symptom Input to Triage Suggestion)**: You successfully tested **GPT-4** to generate triage recommendations from input symptoms.
   * **PoC 2 (Whisper Speech-to-Text Integration)**: You’ve successfully integrated **Whisper** for transcribing audio into text and validated it with sample audio files.
2. **Phase 1: Core Pipeline & Reasoning Flow**:
   * You’ve developed the **core logic** for symptom input processing.
   * You have **implemented the reasoning flow** for generating clarifying questions and triage decisions using GPT-4.
   * The **raw response** for symptoms like fever is being successfully generated and processed.

**Current Status:**

* You are now **ready to focus on**:
  1. **Testing and refining the reasoning flow**: Ensuring that the system works consistently for a variety of symptoms and handles different user inputs (text/audio).
  2. **Building the user interface**: You need to set up an **interactive UI** (via **Gradio** or **Streamlit**) where users can input symptoms and interact with the system for clarifying questions, triage, and advice.

**Next Step:**

The next immediate step is to **complete Phase 1** by implementing the **user interaction flow** (UI) and **refining the reasoning pipeline**.

Here’s a breakdown of the immediate next tasks:

**Immediate Next Tasks (Phase 1 Completion):**

1. **Build the User Interface** (using **Gradio** or **Streamlit**):
   * **Gradio** provides a simple way to create web-based UIs for machine learning models.
   * **Streamlit** is also a great option for building a clean, user-friendly interface for the system.

Example using **Gradio**:

import gradio as gr

def generate\_guidelines(symptom):

# Call your reasoning flow and return the result

guideline = generate\_guideline(symptom)

return guideline

interface = gr.Interface(fn=generate\_guidelines, inputs="text", outputs="text")

interface.launch()

* + **Input**: User enters the symptom (e.g., "Fever").
  + **Output**: Display the clarifying questions, triage options, and safe advice based on the symptom.

1. **Integrate the User Flow**:
   * Ensure that the system processes inputs from both **text** and **speech** (via Whisper) smoothly.
   * Implement the **decision flow** where the system asks follow-up questions based on initial symptom input and recommends appropriate actions (emergency, doctor visit, monitor at home).
2. **Testing the Reasoning Flow**:
   * Use different **test scenarios** (fever, headache, chest pain) to check if the reasoning flow makes correct triage recommendations.
   * Ensure that the system adapts based on user responses (e.g., follow-up clarifying questions should depend on previous answers).
3. **Save Interaction Data**:
   * Log user interactions (symptoms, clarifying questions, responses, triage recommendations) for future improvement.

**Timeline for Next Steps:**

* **Day 1–2**: Build and test the **interactive user interface** (Gradio or Streamlit).
* **Day 3**: **Integrate Whisper** for voice input and test the end-to-end flow with both text and speech.
* **Day 4**: **Test** the system with multiple symptoms and ensure it provides correct triage recommendations and advice.
* **Day 5**: **Refine** the system based on initial testing, and prepare the system for presentation (demo video, report, slides).

Let me know if you need further details on any of these steps, or if you’re ready to proceed with any particular task!

Yes, the description you mentioned for **Phase 1: Core System Implementation** is still accurate and outlines the core steps to move forward. Here's a more **refined plan** for **Phase 1** that aligns with the tasks you have already accomplished and the next steps:

**Phase 1: Core System Implementation**

**Duration**: **Week 1 (Day 3–5)**  
**Objective**: Build the **core system** that processes symptom input, generates clarifying questions, and makes triage decisions.

**Key Tasks in Phase 1:**

**1. Define Symptom Scope:**

* **Task**: Choose **5-7 symptoms** that your system will focus on initially. You’ve already started with **Fever**, but expanding to additional symptoms will allow for a more robust prototype.
* **Symptoms Examples**:
  + Fever
  + Sore throat
  + Headache
  + Chest pain
  + Cough
  + Fatigue
  + Nausea

**2. Gather Reference Guidelines:**

* **Task**: Collect publicly available **triage guidelines** or symptom management guidelines.
  + Example sources: **Mayo Clinic**, **MedlinePlus**, **WebMD**, and **WHO**.
* These guidelines will provide the medical basis for **clarifying questions** and **triage decision-making**.

**Action Items**:

* Download or extract relevant sections from trusted medical resources.
* Use these to **inform the decision-making flow** (i.e., what questions to ask, what conditions to look for, when to recommend a doctor visit or emergency care).

**3. Design GPT-4 Prompt Templates:**

* **Task**: **Design and test prompt templates** for GPT-4 to ensure the model generates consistent and accurate responses.
  + **Clarification**: Use GPT to generate questions like, "How long have you had this symptom?" or "Do you have any other symptoms?"
  + **Triage Decision**: Based on symptom input, GPT should recommend whether the user needs emergency care, should visit a doctor, or can monitor at home.
  + **Safe Advice**: Once the triage decision is made, GPT should generate actionable safe advice for each triage level.

**Action Items**:

* **Define GPT prompt templates** for different symptoms, ensuring that GPT generates the clarifying questions, conditions to flag, triage recommendations, and safe advice in a structured way.
* Test these templates to ensure the results are appropriate.

**Example Prompt** for Fever:

prompt = f"""

For the symptom "{symptom}", provide the following details:

1. 2-3 clarifying questions to ask the patient.

2. Red flag conditions to look for.

3. Triage options (Emergency, Doctor visit, Monitor at home).

4. Safe advice for each triage level (what to do or when to seek care).

Please respond in a structured format with no markdown, extra characters, or code blocks.

"""

**4. Build Frontend Interface:**

* **Task**: Build an **interactive frontend** using **Streamlit** or **Gradio** to collect user input and display recommendations.
* Users will enter a **symptom** and receive a series of **clarifying questions** and triage recommendations.
* The **interactive interface** will:
  + Display **clarifying questions** and allow the user to answer them.
  + Present **triage recommendations** based on user input.
  + Show **safe advice** depending on the triage decision.

**Action Items**:

* Use **Streamlit** or **Gradio** to create a simple UI for symptom input.
  + For example, in **Streamlit**:
  + import streamlit as st
  + symptom = st.text\_input("Enter a symptom:")
  + if symptom:
  + response = generate\_guidelines(symptom)
  + st.write(response)

**5. Test Prototype:**

* **Task**: Test the **core functionality** with mock data to ensure that the system correctly generates triage recommendations and clarifying questions.
* **Action Items**:
  + **Manually test the UI** with various symptoms to ensure everything works as expected.
  + Test with **realistic scenarios**:
    - "I have a fever of 102°F."
    - "My throat hurts and I feel tired."
    - "I have chest pain and shortness of breath."
  + **Debug any issues**: If the questions or recommendations seem inaccurate, adjust the prompt templates or reasoning logic.

**Expected Outcome by End of Phase 1:**

1. **Symptom Input Handling**: Users can input symptoms, and the system responds with clarifying questions, triage recommendations, and safe advice.
2. **Frontend UI**: A functional UI that allows symptom input and displays the results.
3. **Triage Logic**: The system produces relevant and medically sound advice for the symptoms entered.

**Next Step After Phase 1:**

Once **Phase 1** is completed, the system will be able to:

* Take symptoms as input.
* Ask for clarification based on symptoms.
* Provide triage advice (emergency, doctor visit, monitor at home).
* Provide safe advice.

After **Phase 1**, the next step would be to integrate **Whisper** for speech-to-text capabilities (Phase 2), enhance the user interface (Phase 3), and begin testing the complete system with real users.

Let me know if you'd like additional details for any specific part of **Phase 1**, or if you're ready to proceed with the next steps!

Great question! The order of tasks in **Phase 1** depends on the **dependencies** between the tasks. Here's a clearer breakdown to help you decide:

**Option 1: UI First**

Building the **UI** first has its merits. Once you have the **interface** set up, you can start **iteratively** testing the system with user input. However, the **gathering of reference guidelines** will be essential for providing the right answers.

**Pros of Building the UI First:**

1. **Faster Feedback**: As you start building the UI, you'll get immediate feedback on whether the system is producing meaningful and relevant outputs.
2. **User Testing**: You can immediately begin **testing the interface** with mock data to ensure it’s user-friendly and the logic flow is working.
3. **Iterative Process**: You can **build the logic** for symptom processing while simultaneously adjusting the UI.

**Cons:**

1. You may end up with **generic responses** unless you’ve already gathered **medical guidelines** to inform the system.

**Option 2: Gather Reference Guidelines First**

Gathering the **reference guidelines** first is more methodical. This will ensure that the responses your system generates are **medically sound** and **contextually relevant**. Without solid reference guidelines, the system’s output could be **inaccurate** or **generic**.

**Pros of Gathering Guidelines First:**

1. **Accuracy**: You'll ensure that the **clarifying questions** and **triage options** provided by GPT-4 are **medically accurate** and aligned with established practices.
2. **Informed Design**: You can **structure your UI** to reflect the actual triage process and recommendations based on real medical guidelines.
3. **Consistency**: Ensuring the right medical input before testing guarantees **consistent responses** from the system.

**Cons:**

1. The UI work might **delay testing** of the end-to-end system. You’ll need to manually code responses or have mock data until the guidelines are gathered.
2. **Dependency**: You’ll be somewhat constrained by the guidelines while designing the UI.

**Recommendation:**

I suggest **starting with "Gather Reference Guidelines"** first, for the following reasons:

* **Medical Accuracy**: This is the foundation of your project. Ensuring that the clarifying questions and triage recommendations are medically sound should be your first priority.
* **Informed UI Design**: Once you have the **triage guidelines** and **clarifying questions**, you can structure the **UI** more effectively, allowing you to display medically valid options to users.
* **Efficient Testing**: When you have the guidelines in place, you can test the complete system (UI + logic) with actual data, ensuring the system works from both a functional and medical perspective.

**Updated Next Step:**

1. **Gather Reference Guidelines**:
   * Collect publicly available medical triage guidelines for common symptoms.
   * You can pull from trusted resources like **Mayo Clinic**, **MedlinePlus**, or **WHO** to gather accurate guidelines for symptoms like fever, headache, chest pain, etc.
2. **Define GPT-4 Prompt Templates**:
   * Once you have these guidelines, you can **update** the GPT-4 prompt templates, ensuring they align with the established guidelines.
3. **Then Build the UI**:
   * With the reference guidelines in hand, you can proceed to **build the UI**, where the user can enter symptoms, receive clarifying questions, and get triage recommendations.

**Conclusion:**

* Start with **gathering reference guidelines** and defining the **GPT-4 prompt templates**. This will lay the groundwork for building a **medically accurate system**.
* Once those are in place, **build the UI** to interact with the system and test everything together.

Let me know if you’d like more detailed steps on how to gather those guidelines or any additional help with the next task!