Phase 1: Strategic Refocusing - From "What" to "Why"

Goal: Narrow your project's scope from a general "video describer" to a purpose-built "assistive AI for navigation and interaction." Every decision from now on must answer the question: **"How does this directly help a visually impaired person?"**

Step 1.1: Define Core Assistive Use Cases (The "Jobs to be Done")

Instead of describing any random video, focus on 3-5 specific, high-value scenarios for a visually impaired user. This will guide your code, prompts, and evaluation.

- **Scenario 1: Indoor Navigation & Obstacle Avoidance.**
  - **User Goal:** "I'm in my living room. Where is the empty chair? Is there anything in my way?"
  - **Ideal AI Output:** "You are in the living room. There is a clear path ahead. An armchair is to your right, about 5 feet away. A coffee table is to your left."
- **Scenario 2: Object Localization & Interaction.**
  - **User Goal:** "I'm at my desk. Where is my water bottle?"
  - **Ideal AI Output:** "Your water bottle is on the right side of your desk, next to your keyboard."
- **Scenario 3: Environmental Awareness.**
  - **User Goal:** "I've just entered a cafe. What's the general layout?"
  - **Ideal AI Output:** "You are in a cafe. The counter is directly in front of you. There are tables to your left and an open seating area to your right."
- **Scenario 4: Dynamic Hazard Detection.**
  - **User Goal:** "I'm walking on a sidewalk. Is anyone or anything approaching me?"
  - **Ideal AI Output:** "A person is walking towards you, about 10 feet ahead. The path is otherwise clear."

Your Action: Create a new document in `/Documentation/UseCases.md` and formally write these down. These are now your project's guiding principles.

Phase 2: Code Consolidation and Refinement

Goal: Unify your best experimental code into a single, clean, and focused application that directly addresses the use cases from Phase 1.

Step 2.1: Create the Final Application Directory

Your `Software` folder has multiple versions. It's time to consolidate.

1. Create a new folder: `/Software/AIris-Core-System/`.
2. This new folder will house the definitive version of your software. Copy the contents from `/Software/3-Performance-Comparision/` into it, as this is your most advanced prototype.
3. The other folders (`0-Inference-Experimental`, `1-Inference-LLM`, etc.) are now your "archive" of previous work.

Step 2.2: Refactor the Code for Clarity and Purpose

A single `app.py` is great for prototyping but harder to maintain. Let's structure it professionally.

- **`pipeline.py`**: Move the core logic here (frame extraction, BLIP description).
- **`llm_integrations.py`**: Move the Groq and Ollama functions here. This file will handle all interactions with LLMs.
- **`prompts.py`**: Create this new file. Store your system prompts here as constants. This is critical for the next step.
- **`app.py`**: This should now be much cleaner. It will import functions from the other files and manage the Gradio UI.

Step 2.3: Evolve Your Prompt Engineering for Assistive Tasks

Your current "motion analysis expert" prompt is excellent but generic. Now, create specialized prompts based on your use cases.

In `prompts.py`, define multiple system prompts:

```
# prompts.py

NAVIGATION_PROMPT = """
You are an AI assistant for a visually impaired user. Your primary goal is safety and s
Based on the following sequence of visual observations, describe the user's immediate s
Focus on:
1.  Identifying clear paths and potential obstacles (tables, chairs, steps).
2.  Estimating relative positions and distances (e.g., 'to your left', 'in front of you
3.  Describing the general layout of the room.
Your output must be a concise, clear, and actionable description.
"""

OBJECT_FINDER_PROMPT = """
You are an AI assistant helping a visually impaired user find an object.
Based on the visual observations, describe the location of common objects on a surface
Be precise about relative locations (e.g., 'next to the lamp', 'to the right of the boo
Focus only on describing the objects and their placement.
"""
```

In your `app.py`, you can add a dropdown to let the user select a "Mode" (e.g., Navigation, Object Finder), which then uses the appropriate prompt.

Phase 3: Building a Relevant Evaluation Framework

Goal: Create a meaningful way to benchmark your system's performance *as an assistive device*, not just a generic AI.

Step 3.1: Create a Custom Evaluation Dataset

Stop using generic stock videos. They don't represent your use cases.

1. **Record 10-15 Short Videos (10-20 seconds each):** Use your phone at eye-level to simulate a first-person view. Record videos that match your defined use cases (e.g., walk into a room, approach a desk, walk down a hallway).
2. **Create a "Ground Truth" File:** For each video, write down the *ideal assistive description*. This is your benchmark.
    - `video_01.mp4` -> **Ground Truth:** "You are entering a bedroom. A bed is directly in front of you. There is a clear path to the left."

- `video_02.mp4` -> **Ground Truth:** "A person is walking towards you from the end of the hallway."

Step 3.2: Define Your Evaluation Metrics

Your `ollama_performance_report.md` shows raw speed. Your `3-Performance-Comparision` shows semantic similarity. Let's combine and refine these into metrics that matter.

1. **Latency (Quantitative):** Time from input to final description. (You already have this). **Target: < 2 seconds.**
2. **Semantic Helpfulness Score (Quantitative):** Use the cosine similarity from your `3-Performance-Comparision` code, but compare the AI's output against your new, high-quality "Ground Truth" descriptions. **Target: > 0.85 similarity.**
3. **Task Success Rate (Qualitative):** For each video, ask: "Does the AI's description enable the user to complete the intended task?" (e.g., find the chair, avoid the obstacle). Score this as Yes/No. **Target: > 90% "Yes".**
4. **Safety Criticality Score (Qualitative):** Did the AI correctly identify and prioritize hazards? (Score 0=Missed Hazard, 1=Mentioned Hazard, 2=Prioritized Hazard).

Step 3.3: Document Your Findings

Create a new document: `/Documentation/EvaluationReport.md`. Present your results in a clear table for your presentation.

Video ID	Latency (s)	Semantic Score	Task Success	Safety Score	Best Performing LLM
`indoor_nav_01.mp4`	1.8s	0.91	Yes	2	`llama-3.1-70b`
`object_find_01.mp4`	1.6s	0.88	Yes	N/A	`gemma2-9b`

Phase 4: Crafting the Presentation Narrative

Goal: Tell a compelling story that frames your technical work as a solution to a real human problem.

Presentation Outline (10-12 Slides, ~15 Minutes):

1. **Title Slide:** Alris: AI That Opens Eyes. (Your names, course).

2. **The Problem:** Start with empathy. "For millions of visually impaired individuals, simple tasks like navigating a room or finding an object are daily challenges." Show a statistic. Briefly mention the limitations of current solutions (canes, apps).
3. **Our Vision:** "We introduce AIris, a wearable AI system designed to provide real-time, contextual awareness, bridging the gap between the user and their environment."
4. **The Core Challenge:** "How do we go from a simple image to an *actionable, safe, and instant* description?" This frames your technical work.
5. **Our System Architecture:** A high-level diagram. (Video Frame -> **Local Vision Model (BLIP)** -> Observation Text -> **LLM Reasoning Engine (Groq)** -> Assistive Description). Explain WHY this hybrid approach is smart (local for speed/privacy, LLM for intelligence).
6. **Innovation in Action: Task-Specific Prompting:** This is your key technical contribution.
    - Show a generic prompt's output (e.g., "a room with a chair").
    - Show your `NAVIGATION_PROMPT` 's output ("A chair is to your right...").
    - Explain that you've engineered the AI's "brain" to think like an assistant, not a describer.
7. **Live Demo / Video: This is the most important slide.**
    - Show a screen recording of your final Gradio app.
    - Use one of your custom-recorded videos (e.g., `indoor_nav_01.mp4`).
    - Show the video, the "Ground Truth" you wrote, and then run your pipeline to show the AI's live output.
8. **Evaluation & Results:** Show the `EvaluationReport.md` table.
    - "We tested AIris on a custom dataset of 15 real-world scenarios."
    - "Our system achieved an average latency of **1.7s**, a semantic helpfulness score of **0.89**, and a **93% task success rate**." This is powerful and proves your system works.
9. **Current Status & Next Steps (499B):**
    - "We have a validated, high-performance software core."
    - "Our next steps are to integrate this software into our prototype hardware (Raspberry Pi 5) and begin user testing to refine the experience."
10. **Conclusion:** Briefly summarize the problem, your innovative solution, and the impact AIris can have. End with your tagline: "AI That Opens Eyes."
11. **Q&A**

By following this plan, you will transform your impressive collection of technical experiments into a single, powerful, and well-documented system that is perfectly aligned with your project's mission. Your presentation will be focused, data-driven, and tell a compelling story of solving a real-world problem.