### Product Requirements Document (PRD) - SIMIS.AI

**1. Introduction**

* **Project Name:** SIMIS.AI
* **Vision:** To provide real-time, interactive, and personalized guidance for using medical devices, empowering patients for safer and more confident self-care.
* **Target Audience:** Patients who need to use medical devices at home and may have forgotten instructions from a healthcare professional.
* **Problem Statement:** Patients often forget or misremember how to use medical devices correctly after receiving one-time instructions, leading to incorrect usage and potential health risks. Existing resources like manuals and videos are passive and cannot provide real-time corrective feedback.
* **Solution:** A web app optimized for desktop that uses computer vision and an LLM to provide live, step-by-step guidance on how to use a medical device, with immediate corrective feedback and multi-language support.

**2. Functional Requirements (MVP)**

* **FR-1: Real-time Video Stream:** The web app must display a live video feed from a connected camera.
* **FR-2: Product Detection:** The app must be able to detect and identify a medical product in the video stream.
* **FR-3: User Interaction Detection:** The app must be able to detect key user actions with the product, such as hand gestures, finger placement, and button presses.
* **FR-4: Dynamic Guidance:** The app must provide step-by-step instructions in both text and verbal formats based on the user's progress and actions.
* **FR-5: Corrective Feedback:** The app must provide immediate verbal and/or text feedback when a user performs an incorrect action.
* **FR-6: Multilingual Support:** The app must be able to provide guidance in various Southeast Asian languages.
* **FR-7: Instruction Data Source:** The app must be able to parse instructions from product inserts (via OCR), PDF manuals, or online search.
* **FR-8: User Interface:** A clean and intuitive UI with a live video feed, a chat/text box for instructions, and controls for starting/stopping the process.

**3. Non-Functional Requirements**

* **NFR-1: Performance:** The app must provide real-time feedback with minimal latency.
* **NFR-2: Scalability:** The tech stack must be scalable to accommodate a growing number of users and medical products.
* **NFR-3: Usability:** The app should be easy to use for patients of varying technical abilities.

### Optimized Tech Stack for SIMIS.AI (MVP)

* **Frontend:** Astro + React islands (with Tailwind CSS for rapid styling)
* **Backend:** Vercel Edge Functions or Netlify Functions for POC; AWS Lambda for production deployment.
* **Computer Vision:** MediaPipe Web & TensorFlow.js
* **LLM Integration:** Direct API calls to **SEA-lion**
* **LLM Functionality:** Retrieval-Augmented Generation (RAG)
* **Speech Technologies:** Text-to-Speech (TTS) and Speech-to-Text (STT) for voice interactions.
* **Storage:** Supabase (for user data, if needed) & local storage (for preferences).
* **Data Sourcing:** OCR library (e.g., Tesseract.js), PDF parsing library, Web scraping tools.
* Cloud Infrastructure & Deployment: Vercel or Netlify for rapid POC deployment; AWS (S3 for hosting, Lambda for serverless functions, DynamoDB for database) for production.

### Project Outline Plan (1-Week POC & Future-Proofing)

**Team Members & Roles:**

* **Jenxi:** Front-end Development (UI/UX, Astro, React, Tailwind CSS)
* **Raymond:** Computer Vision & AI (MediaPipe, TensorFlow.js, CV model training)
* **Full Stack/Deployment (You):** Full-stack Development, Deployment, LLM Integration (SEA-lion, RAG), and Data Sourcing.

**Day 1-2: Setup & Foundation**

* **All:** Finalize the POC scope (e.g., a single product like a thermometer).
* **Jenxi:** Set up Astro/React project. Build a single page with a live camera feed and a placeholder for text instructions.
* **Raymond:** Create a simple proof-of-concept for camera access using WebRTC. Begin training a basic object detection model for the chosen product.
* **Full Stack (You):** Set up the Vercel/Netlify deployment pipeline. Integrate a basic API call to SEA-lion to test its response.

**Day 3-4: Core Integration**

* **All:** Daily sync-ups to ensure components are compatible.
* **Jenxi:** Develop the UI to receive and display dynamic text from the LLM based on CV events.
* **Raymond:** Integrate the product detection model with Jenxi's front end. Develop a simple state-based CV logic (e.g., "product detected," "incorrect position").
* **Full Stack (You):** Connect Raymond's CV output (e.g., a JSON message with the detected state) to the SEA-lion API call. Create a basic prompt template that includes the CV state and a request for a multilingual response. Implement a basic TTS function to read out the LLM's response.

**Day 5-6: Refinement & Testing**

* **All:** End-to-end testing of the core loop: Camera -> CV -> LLM -> Text/Voice.
* **Jenxi:** Refine the styling and UI flow for a seamless demo.
* **Raymond:** Fine-tune the CV model to reduce false positives/negatives for the specific product.
* **Full Stack (You):** Polish the SEA-lion prompt for better multilingual output. Ensure the TTS integration is smooth. Finalize the web app deployment for a public-facing URL.

**Day 7: Finalization & Submission**

* **All:** Final bug fixes and demo preparation.
* **Full Stack (You):** Prepare the project proposal, the one-liner for the tech stack, and a clear demo video showing the core functionality.

Phase 5: Productionization & AWS Migration

* All: Plan for the full production environment.
* Full Stack (You):
  + Refactor backend functions to be deployed on AWS Lambda.
  + Set up AWS S3 for hosting static frontend assets.
  + Configure AWS API Gateway to route API calls.
* Raymond:
  + Scale and optimize CV models for production.
* Jenxi:
  + Prepare the frontend for a static site build and deployment to S3.