AI-Powered Triage App: Minimum Viable Product (MVP) Product Requirements Document

1. Project Title (MVP Focus)

AI-Powered Patient Triage Demo App

2. Executive Summary (MVP)

This MVP aims to demonstrate the core concept of an AI-powered patient intake system. It will showcase a streamlined process where a patient inputs their primary complaint, the system uses an LLM to ask a few adaptive follow-up questions, and then generates a basic, physician-ready "SOAP-like" note. The primary objective is a visually appealing, functional proof-of-concept for a 5-minute demo, emphasizing the potential for efficiency and improved data collection.

3. Problem Statement (MVP)

Current patient intake often involves manual, repetitive questioning and can lead to inconsistent data. This MVP addresses the initial pain point of inefficient patient data gathering by demonstrating how an AI can intelligently collect pertinent information, reducing administrative burden and setting the stage for more focused physician interactions.

4. Project Goals (MVP)

- To create a visually appealing, interactive demo of an AI-powered patient triage flow.
- To demonstrate the use of an LLM for natural language understanding and adaptive questioning for a single, predefined symptom path.
- To generate a simplified, structured output (SOAP-like note) based on patient input.
- To showcase basic data persistence for physician review.
- To achieve these goals within a few weeks on a tight budget, prioritizing demo-readiness over production-grade robustness.

5. Key Features (MVP - Demo Focused)

5.1. Patient Complaint Input

- **Description:** A simple, prominent text input field where a patient can type their primary complaint (e.g., "I have a headache," "My throat hurts").
- **Interaction:** User types, then clicks a "Start Triage" button.

5.2. Simplified AI-Powered Adaptive Questioning

- **Description:** Upon receiving the initial complaint, the system will use an LLM (e.g., Gemini API) to generate a sequence of 2-3 highly relevant follow-up questions. For the demo, this will focus on a *single, pre-selected symptom category* (e.g., "Headache" or "Sore Throat") to simplify the LLM's prompt and expected output.
- **Interaction:** Each question appears one at a time. The user types their response, and the system processes it before presenting the next question.
- **Edge Cases:** The LLM will be prompted to handle simple conversational nuances and provide a default "I don't understand" if input is completely irrelevant, without complex error handling.

5.3. Basic SOAP-like Note Generation

- **Description:** After the adaptive questioning, the LLM will synthesize the collected patient responses into a very basic, structured "SOAP-like" note.
    - **S (Subjective):** A summary of the patient's chief complaint and key symptoms in their own words.
    - **O (Objective):** A placeholder or a simple statement like "No objective data collected by app."
    - **A (Assessment):** A very high-level, AI-generated "possible" assessment (e.g., "Possible tension headache"). Emphasize this is *not* a diagnosis.
    - **P (Plan):** A generic suggestion like "Recommend physician review for further assessment."
- **Interaction:** The generated note is displayed to the patient/user.

5.4. Basic Data Storage & Physician Review

- **Description:** The generated SOAP-like note and the raw patient responses will be stored in a simple, non-relational database (e.g., Google Cloud Firestore). A separate, very basic "Physician Review" screen will display a list of submitted notes, allowing a physician (or demo presenter) to click on one and view its content.
- **Interaction:** A "View Notes" button on the main screen leads to a list. Clicking a note displays its details.

6. User Flow (Simplified)

1. **Landing Page:** User sees a clear input field for their primary complaint.

2. **Complaint Submission:** User types complaint and clicks "Start Triage."
3. **Adaptive Questioning:**
    - System displays Question 1. User types response.
    - System displays Question 2 (based on LLM processing of previous response). User types response.
    - System displays Question 3. User types response.
4. **Note Generation & Display:** System displays the generated SOAP-like note.
5. **Physician Review (Separate Screen):**
    - User navigates to a "Physician Review" section.
    - A list of previously submitted notes is displayed.
    - Clicking a note shows its full content.

7. Technology Stack (Budget-Friendly & Quick Dev)

- **Frontend:** HTML, CSS (Tailwind CSS for rapid styling), JavaScript. This allows for quick development and deployment.
- **Backend/LLM Integration:** Direct API calls from frontend JavaScript to the Gemini API for text generation. This avoids the need for a separate backend server for the MVP.
- **Data Storage:** Google Cloud Firestore (NoSQL database). This offers a generous free tier, easy setup, and real-time updates for the physician review.
- **Deployment:** Simple static hosting (e.g., Firebase Hosting, GitHub Pages) for the HTML/CSS/JS.

8. Success Metrics (Demo Focused)

- **Visual Appeal:** The app looks polished and professional for a demo.
- **Smooth Flow:** The transition from complaint input to questioning to note generation is seamless.
- **LLM Responsiveness:** The LLM provides relevant and quick responses for the demo path.
- **Note Readability:** The generated SOAP-like note is easy to understand and demonstrates the concept.

- **Budget Adherence:** Development costs remain minimal, utilizing free tiers of services.
- **Completion within Timeline:** The demo is ready within the few-week timeframe.

9. Non-Goals for MVP (Explicitly Out of Scope)

- **Robust Security:** Production-level authentication, authorization, and data encryption.
- **HIPAA/GDPR Compliance:** This is a demo, not a clinical tool.
- **Extensive Questionnaires/Logic Trees:** Only a single, short adaptive path will be implemented.
- **Complex AI Training:** No custom AI model training; reliance on pre-trained LLM (Gemini API).
- **Voice Input/Multi-language Support:** Text input only, English only.
- **Integration with EHRs or Diagnostic Tools:** Standalone application.
- **Comprehensive Error Handling:** Basic error messages for API failures.
- **User Accounts/Profiles:** All data is anonymous or tied to a generic session for demo purposes.
- **Mobile App Native Development:** Web-based only.

10. Timeline (Aggressive for POC - Target: 2-3 Weeks)

- **Week 1:**
    - **Day 1-2:** Frontend UI/UX design (wireframes, basic styling with Tailwind).
    - **Day 3-4:** Implement Patient Complaint Input and initial display of questions.
    - **Day 5:** Integrate Gemini API for initial question generation (hardcode prompt for specific symptom).
- **Week 2:**
    - **Day 1-2:** Refine adaptive questioning logic (simple if/else based on LLM response keywords).
    - **Day 3-4:** Implement SOAP-like note generation using LLM.
    - **Day 5:** Set up Firestore and implement basic saving of notes.

- **Week 3:**
    - **Day 1-2:** Develop Physician Review interface to display notes from Firestore.
    - **Day 3-4:** Polish UI/UX, add animations/transitions for smooth demo.
    - **Day 5:** Testing, bug fixing, and preparation for demo.

11. Budget Considerations

- **LLM API:** Utilizing the Gemini API's free tier or low-cost usage.
- **Database:** Firestore's generous free tier.
- **Hosting:** Free static site hosting options.
- **Development:** Focus on rapid "vibe coding" with readily available libraries and frameworks to minimize development time and associated costs.