Candidate Take-Home Challenge

Note to Candidate: This exercise is designed to take approximately 5–7 hours in total. It evaluates your ability to design and orchestrate LLM-powered reasoning agents that combine structured patient data with unstructured clinical guidelines, and to reuse a single RAG pipeline across both deterministic decision support and conversational clinical querying.

Part 1: The "NG12 Cancer Risk Assessor"

1. The Objective

Build a Clinical Decision Support Agent using **Google Vertex AI (Gemini 1.5)**. The agent will accept a Patient ID, retrieve their records, consult the **Official NICE NG12 Cancer Guidelines** (PDF), and output a risk assessment with citations.

2. The Architecture

The solution must be a **FastAPI** service (wrapped in Docker) that follows this logic flow:

1. **User Input:** Receives Patient ID via API.
2. **Tool Use (Data Retrieval):** Agent calls a tool to fetch structured data (Age, Symptoms) from a simulated database.
3. **RAG (Guideline Lookup):** Agent searches a Vector Store for relevant sections of the **Full NG12 PDF** based on the patient's symptoms.
4. **Reasoning:** Agent synthesizes the data to determine if the patient meets the criteria for "Urgent Referral" or "Urgent Investigation."
5. **Output:** Returns a JSON response with the assessment and specific guideline citations.

3. The Data Package

A. Structured Data (Save as patients.json)
This simulates your BigQuery Table. Your application should ingest this file to mock database retrieval.

JSON


```
[
 { "patient_id": "PT-101", "name": "John Doe", "age": 55, "gender": "Male", "smoking_history": "Current Smoker", "symptoms": ["unexplained hemoptysis", "fatigue"], "symptom_duration_days": 14 },
 { "patient_id": "PT-102", "name": "Jane Smith", "age": 25, "gender": "Female", "smoking_history":
```

"Never Smoked", "symptoms": ["persistent cough", "sore throat"], "symptom_duration_days": 5 },
{ "patient_id": "PT-103", "name": "Robert Brown", "age": 45, "gender": "Male", "smoking_history": "Ex-Smoker", "symptoms": ["persistent cough", "shortness of breath"], "symptom_duration_days": 28 },
{ "patient_id": "PT-104", "name": "Sarah Connor", "age": 35, "gender": "Female", "smoking_history": "Never Smoked", "symptoms": ["dysphagia"], "symptom_duration_days": 21 },
{ "patient_id": "PT-105", "name": "Michael Chang", "age": 65, "gender": "Male", "smoking_history": "Ex-Smoker", "symptoms": ["iron-deficiency anaemia", "fatigue"], "symptom_duration_days": 60 },
{ "patient_id": "PT-106", "name": "Emily Blunt", "age": 18, "gender": "Female", "smoking_history": "Never Smoked", "symptoms": ["fatigue"], "symptom_duration_days": 30 },
{ "patient_id": "PT-107", "name": "David Bowie", "age": 48, "gender": "Male", "smoking_history": "Current Smoker", "symptoms": ["persistent hoarseness"], "symptom_duration_days": 45 },
{ "patient_id": "PT-108", "name": "Alice Wonderland", "age": 32, "gender": "Female", "smoking_history": "Never Smoked", "symptoms": ["unexplained breast lump"], "symptom_duration_days": 10 },
{ "patient_id": "PT-109", "name": "Tom Cruise", "age": 45, "gender": "Male", "smoking_history": "Never Smoked", "symptoms": ["dyspepsia"], "symptom_duration_days": 7 },
{ "patient_id": "PT-110", "name": "Bruce Wayne", "age": 60, "gender": "Male", "smoking_history": "Never Smoked", "symptoms": ["visible haematuria"], "symptom_duration_days": 2 }
]

B. Unstructured Data (The PDF)

- **Action Required:** Download the full guideline PDF: **"Suspected cancer: recognition and referral (NG12)"**
- **Download URL:**

  https://www.nice.org.uk/guidance/ng12/resources/suspected-cancer-recognition-and-referral-pdf-1837268071621

- **Instruction:** You must build a pipeline to **parse this PDF**, create embeddings (using Vertex AI Embeddings or compatible), and store them in a local Vector Database (e.g., ChromaDB, FAISS) for the agent to query.

4. Requirements & Deliverables

Please submit a link to a **GitHub Repository** containing:

1. **Source Code:** Modular Python code.
   - **PDF Ingestion Script:** A script that parses the NG12 PDF and builds the vector index.
   - **Agent Logic:** The reasoning engine using Gemini 1.5.
   - **Tooling:** Function calling to retrieve patient data.
2. **Prompt Engineering:** A Markdown file (PROMPTS.md) explaining your System Prompt strategy.
3. **Dockerfile:** A working configuration to build the service.

4. **UI:** A minimal frontend to input the Patient ID and view the result.

Part 2: Conversational AI Over the NG12 Knowledge Base (Chat Mode)

The Objective

Extend your solution by adding a **conversational agent** that can answer questions using the **same NG12 PDF content already embedded and stored in your vector database** (built in Part 1). This tests your ability to reuse your RAG pipeline in a multi-turn setting, handle citations, and maintain grounded responses.

Core Capability

Build a **chat interface + API** where a user can ask questions such as:

- "What symptoms trigger an urgent referral for lung cancer?"
- "Does persistent hoarseness require urgent referral, and at what age?"
- "What does NG12 say about dyspepsia and thresholds for investigation?"
- "Summarize the referral criteria for visible haematuria."

The agent must:

1. **Retrieve** relevant guideline chunks from the existing vector store.
2. **Answer** in natural language using only retrieved evidence.
3. **Cite** the specific guideline source passages (page/section identifiers from the PDF ingestion pipeline).

Requirements

A. New API Endpoints (FastAPI)

Add endpoints that support multi-turn conversation:

1. POST `/chat`

- **Input JSON**:
  - `session_id` (string) – required (client-generated is fine)
  - `message` (string) – required
  - `top_k` (int, optional) – default e.g. 5
- **Behavior**:
  - Perform RAG against the NG12 vector store.

    - Return an answer grounded in retrieved passages with citations.
2. `GET /chat/{session_id}/history` *(optional but encouraged)*
- Returns conversation history for the session.
3. `DELETE /chat/{session_id}` *(optional but encouraged)*
- Clears stored history.

B. Conversation Memory

You may store conversation state:

- In-memory (acceptable for the take-home)
- Or in a lightweight store (SQLite/Redis) if preferred

Memory must be used to support follow-ups like:

- "What about if the patient is under 40?"
- "Can you quote the part about duration thresholds?"

C. Grounding & Guardrails

The chat agent must:

- **Refuse or qualify** answers if retrieval returns insufficient evidence ("I couldn't find support in the NG12 text for that…").
- Avoid inventing thresholds or criteria not supported by retrieved chunks.
- Always include citations when making clinical pathway statements.

D. UX

Extend the minimal UI to include a **Chat tab**:

- A chat message window
- A text input box
- Display citations in a readable format (e.g., `[NG12 p.123]` with a short excerpt)

Expected Output Format (Chat)

Return JSON like:

```
{
  "session_id": "abc123",
  "answer": "…",
  "citations": [
```

```
    {
      "source": "NG12 PDF",
      "page": 123,
      "chunk_id": "ng12_0123_04",
      "excerpt": "…"
    }
  ]
}
```

Evaluation (Part 2)

The conversational agent will be graded on:

- **Groundedness**: answers consistently supported by retrieved guideline text
- **Citation quality**: cites specific and relevant passages (not generic)
- **Multi-turn coherence**: handles follow-ups using conversation context
- **Failure behavior**: appropriately says "not found / unclear" when evidence is missing
- **Reuse of pipeline**: uses the same vector DB and ingestion outputs from Part 1 (no re-downloading/re-embedding per chat request)

Additional Deliverables (Part 2)

Add to the GitHub repo:

- `CHAT_PROMPTS.md` (or extend `PROMPTS.md`) describing your chat system prompt and grounding strategy
- Any additional UI components for the chat experience
- Brief README section describing how to run chat mode locally and in Docker