



Technical Assignment – Agentic RAG-Powered Referral Matcher

 **Objective:** Design and implement a **Referral Matching AI Agent** that uses a RAG pipeline to recommend appropriate specialists based on patient input. The output should include ranked specialists and reasoning behind the selection (explainability).


 **Scenario:** You are tasked with building a **Specialist Search Agent (SSA)** for the WeKare360 One-Click Referral system. A patient visits a PCP and presents the following: "I'm having chest pain during mild activity, it gets worse with exertion, and sometimes I feel dizzy. The Referral Matching Agent must:

- Use the above query (or variants of it) to search a mock specialist knowledge base.
- Recommend top 3 specialists (e.g., Cardiologist, Pulmonologist) with justification.
- Score them based on proximity, availability, and insurance network (provided in metadata).
- Output a structured JSON response that can be passed to downstream agents like the Insurance Authorization Agent.

What You Need to Deliver

1. **Brief Notebook / Script (Python)**
 - 1.1. Ingest a sample CSV/JSON file with provider metadata (e.g., name, specialty, location, insurance).
 - 1.2. Use a **RAG pipeline** (LangChain, LlamaIndex, or Haystack) over provided unstructured mock specialist profiles.
 - 1.3. Parse a patient's free-text input and perform:
 - 1.3.1. Medical intent extraction (symptoms -> specialty).
 - 1.3.2. Semantic search on profiles.
 - 1.3.3. Rank & justify specialist matches.
2. **Agent Design**
 - 2.1. Describe the logic of the SSA agent.
 - 2.2. Include how it integrates with a Referral Supervisor Agent and the Insurance Auth Agent in a multi-agentic pipeline.
 - 2.3. How would you enable dynamic updates (e.g., real-time availability API)?
3. **Bonus (Optional but valued)**
 - 3.1. Add a reflection or corrective loop: if top 3 results are low confidence (e.g., <0.6 relevance), re-query using a rephrased prompt.
 - 3.2. Include patient-friendly explanation for why each doctor was matched.

Tech Stack Expectations

1. You can use:
 - 1.1. **Python**, Jupyter notebook or script
 - 1.2. Vector store: **FAISS, Chroma, Pinecone** (any of these is fine)
 - 1.3. Embeddings: OpenAI, HuggingFace models
 - 1.4. RAG tooling: LlamaIndex,
 - 1.5. Open-source LLMs or GPT-4-turbo (your choice)
 - 1.6. Clean, modular code
2.  **Timeline & Submission**
 - 2.1. **Time required:** ~4–6 hours
 - 2.2. **Submit:** GitHub repo or zip file with code + 1-pager README
 - 2.3. Include short Loom (or markdown) describing your pipeline, agent behavior, and design decisions (2–3 mins max).