Applied Generative AI Specialisation

Capstone Problem Statement

Agentic Healthcare Assistant for Medical Task Automation

Problem scenario:

In the modern digital health ecosystem, managing patient care involves a range of complex and interdependent tasks. These include appointment scheduling, maintaining patient histories, and retrieving timely disease-related information. Often, healthcare systems rely on siloed tools with limited automation, leading to inefficiencies and fragmented patient experiences.

Agentic AI systems, especially those combining Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG), offer the potential to transform healthcare administration by enabling autonomous coordination of such tasks.

Project objective:

This project aims to develop an Agentic Healthcare Assistant that functions as a virtual medical assistant. The system will utilize Agentic AI frameworks, RAG pipelines, and memory modules to:

- Book medical appointments: Automate slot discovery and scheduling based on patient intent and doctor availability
- Manage medical records: Enable attendants to add or update structured/unstructured patient history
- Retrieve medical histories: Summarize past diagnoses, treatments, and relevant alerts using LLMs
- Perform medical information searches: Fetch up-to-date disease information using trusted external sources such as Medline and WHO

Steps to follow:

This capstone is structured into two parts, covering the Agentic System Design and LLMOps with Streamlit UI:

Part 1: Agentic Healthcare Assistant System Design

1. Agent Planning and Goal Decomposition

- Use a planner to interpret multi-step patient queries
- Break down complex requests into sequential sub-goals
- Identify appropriate tools or APIs to fulfill each task

2. Tool and Memory Setup

- Integrate APIs for:
  - Appointment booking (Doctor Schedule API)
  - Medical history management (EHR DB or Patient DB)
  - Disease search (Web Search APIs, e.g., Bing Search, Medline)
- Use a vector database (e.g., FAISS) to store and retrieve patient summaries
- Configure memory modules to retain long-term patient context

3. Prompt Engineering and Task Chaining

- Create structured prompts tailored to each agentic sub-task
- Design prompt chains that guide LLMs through summarization, planning, and action triggering
- Incorporate patient context in prompts using memory lookups

4. Agent Execution Flow (Sample Scenario)

User Input:
"My 70-year-old father has chronic kidney disease. I want to book a nephrologist for him. Also, can you summarize latest treatment methods?"

Agent Workflow:

1. Identify patient and context
2. Retrieve father's medical history
3. Query doctor calendar and book appointment
4. Search and summarize treatment options via RAG pipeline

Part 2: LLMOps (Model Evaluation, Monitoring, and Streamlit UI)

6. Model Evaluation

- Use QAEvalChain or equivalent to assess accuracy and relevance of generated summaries and search results
- Log and analyze agent performance per module (e.g., success rate of bookings, response precision)

7. Data Visualization and UI

Build a **Streamlit** dashboard that enables:

- Patient and doctor views
- Real-time appointment tracking
- Summary of latest retrieved medical information
- Evaluation metrics for model responses and tool success

8. Memory and Logs Interface

- Display agent memory traces and planning breakdowns
- Include interactive elements to test different user scenarios
- Log tool usage and success/failure across tasks