Task 1: Defining your Problem and Audience

1. Succinct 1-sentence problem description

There is a shortage of expert entry-level physical therapists because current exam prep solutions fail to provide personalized support, causing 20% of candidates to fail their expensive licensure exams and leaving others with unaddressed knowledge gaps.

---

2. 1–2 paragraphs on why this is a problem for the user

Physical therapy licensure candidates face expensive, high-stakes exams that determine when they can begin their careers. Current prep systems focus on broad coverage rather than personalized support, leaving students' specific weaknesses unaddressed. Even those who manage to pass may still carry gaps in knowledge or clinical reasoning skills that can undermine their confidence and performance as new therapists. For the 20% who fail, the consequences are even greater—costly retake fees, additional prep expenses, and a delay of up to a year in entering the profession.

Students need a preparation system that identifies and strengthens their individual weak points rather than cycling them through the same generalized material. Without this, they risk wasting time and money while still starting their careers at a disadvantage, with knowledge gaps that could have been addressed before entering practice.

Task 2: Propose a Solution

☑ **Deliverables**

1. Write 1-2 paragraphs on your proposed solution. How will it look and feel to the user?

   The proposed solution is an AI-driven, adaptive learning platform that continuously assesses each candidate's knowledge gaps and delivers targeted content via NPTE MCQAs in real time. As soon as a student struggles with a concept—say, differential diagnosis of musculoskeletal injuries—the system retrieves and surfaces a short, clinically focused review, generates a few practice questions, and walks them through the reasoning step by step. The interface feels like a personalized tutor: clean dashboards highlight mastery levels by topic, quick "micro-lessons" adapt to the student's pace, and periodic summary reports pinpoint progress and remaining weak spots. For this challenge I will implement only the MVP.

Under the hood, the platform uses a standard LLM-application stack: a vector database holds indexed exam prep material, an agent fetches relevant specific information from the web, a retrieval agent pulls relevant chunks, and an LLM-powered engine generates explanations, practice items, and concise feedback. Behind the scenes, analytics monitor engagement and success rates, so students spend less time on what they already know and more time mastering challenging areas. The result is faster study cycles, lower retake rates, and greater confidence—delivering entry-level therapists who are not only exam-ready but truly practice-ready.

2. Describe the tools you plan to use in each part of your stack. Write one sentence on why you made each tooling choice.

    1. LLM – Claude Opus 4: Offers state-of-the-art performance on complex, agent workflows, delivering superior reasoning accuracy. (That is for my customers). But for this challenge I will probably use the 4.1

    2. Embedding Model – text-embedding-3-small
    Launched January 25 2024, it boosts retrieval benchmarks

    3. Orchestration – LangChain/LangGraph Supports loops, branching, and built-in persistence for managing complex, stateful LLM pipelines with fine-grained control

    4. Vector Database – Qdrant: Delivers ultra-fast indexing, high throughput, and low latency for scalable vector similarity search

    5. Monitoring – LangSmith: Offers unified testing and observability for LLM applications, turning real user data into actionable evaluation and error-tracking insights

    6. Evaluation – RAGAS: Provides a reference-free suite of metrics to assess both retrieval and generation quality in RAG systems, accelerating evaluation cycles without human annotations

    7. User Interface – React: Its component-based architecture and virtual DOM enable reusable, performant UI components for building interactive, maintainable front ends

3. Where will you use an agent or agents? What will you use “agentic reasoning” for in your app?

   My agent will assess whether the retrieved context is of sufficient quality to generate a meaningful question. If the context is inadequate, the agent will perform a web search to supplement it with higher-quality information.

   Using an agent has to reason what knowledge is lacking (based on the student’s mistakes). The retriever will provide explanatory answers, will retrieve knowledge material, and generate the new questions on the topic.

Flow:

MVP Requirements (for the demo):

1. **Frontend (React):**
   - Simple UI for the user to request a MCQ on an NPTE topic There are nine topics present in the exam material.
   - Enter a topic/question (e.g., “neuromuscular issues”)
   - Receive an MCQ from the LLM
   - Select an answer (A, B, C, or D)
   - Receive correct answer and feedback (and explanation correct/incorrect, review materials on the topic to close knowledge gaps, and ask for new MCQs.
   - Using **Vite** in frontend.
2. **Backend (FastAPI + LangChain/LangGraph):**
   - 1st time: LLM is fed new research material from PDF sources.
   - While in session:
     - Receives the user’s topic/question.
     - Uses LangChain/LangGraph to:
     - Generate an NPTE-style MCQ using the LLM (gpt-3.5-turbo for the demo).
     - Agent check’s the user’s answer, decides for correctness.
     - If incorrect, uses web-tool to fetch relevant learning material.
     - Loop until mastery or user changes topic, or user stops.

Stack for MVP

- **Frontend:**
- Vite + React + TypeScript
- **Backend:**
  - FastAPI (Python)
  - Qdrant (vector DB
  - LangChain/LangGraph ,
  - retrieval, Cohere reranker)
- **LLM:** gpt-3.5-turbo
- **Other tools:**
  - LangSmith for trace
  - RAGAS evaluation

Steps

1. Set up the Vite + React frontend

- Scaffold a new Vite project.
- Build a simple UI for the MCQ loop.

2. Set up the FastAPI backend

- Add endpoints for:
- Submitting a topic/question.
- Submitting an answer and getting feedback/new MCQ.
- Integrate with LangChain/LangGraph and (optionally) Qdrant.

3. Connect frontend and backend

- Use fetch in React to call your FastAPI endpoints.

Task 3: Dealing with the Data

You are an AI Systems Engineer. The AI Solutions Engineer has handed off the plan to you. Now *you must identify some source data* that you can use for your application.

Assume that you'll be doing at least RAG (e.g., a PDF) with a general agentic search (e.g., a search API like Tavily or SERP).

Task 3: Collect data for (at least) RAG and choose (at least) one external API

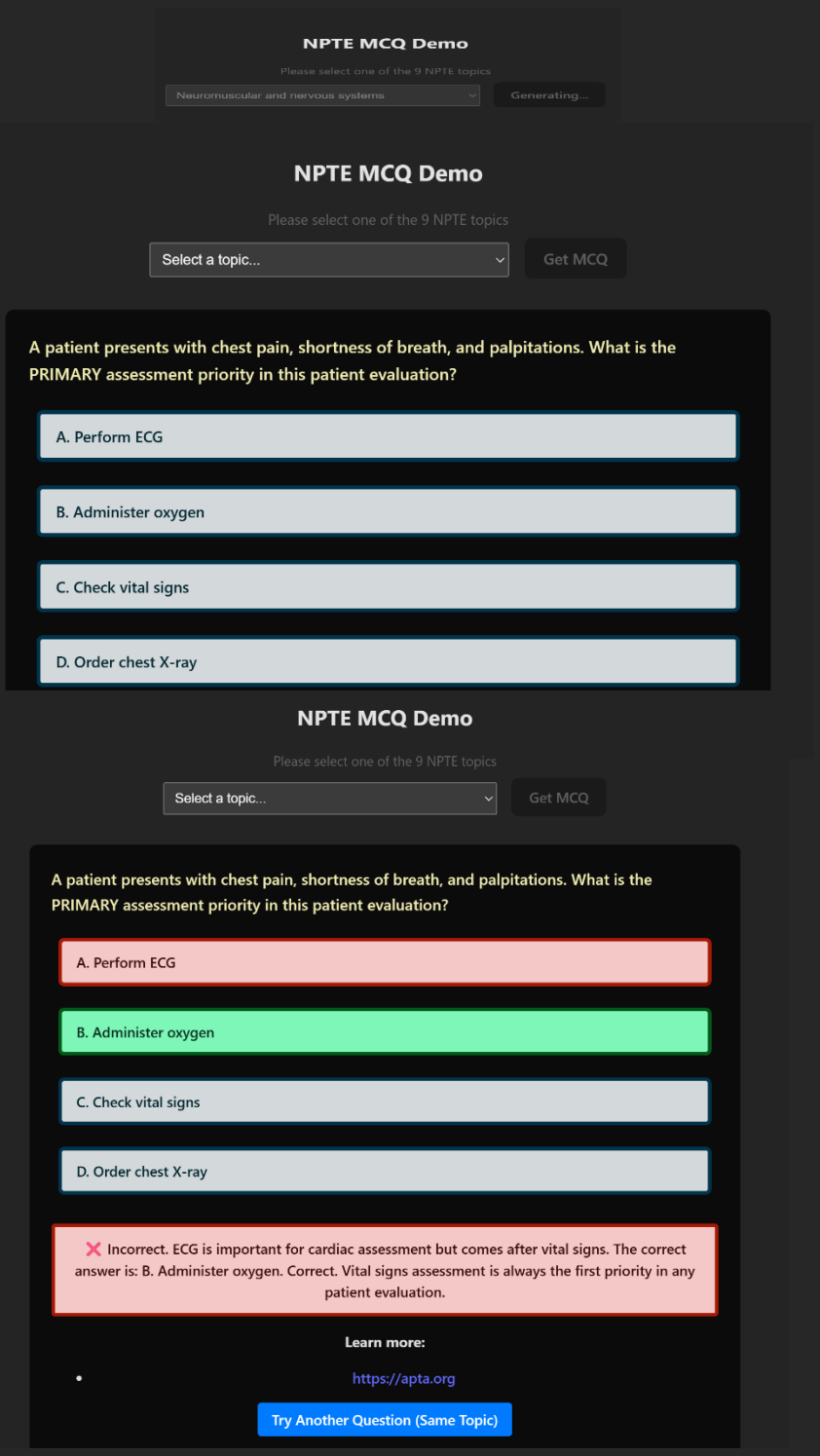
Hint:

- *Ask other real people (ideally the people you're building for!) what they think.*
- *What are the specific questions that your user is likely to ask of your application?* ***Write these down.*** </aside>

☑ **Deliverables**

1. Describe all of your data sources and external APIs, and describe what you'll use them for.
    1. 40 PDFs Google Scholar, NIH, APTA, NPTE-PT exam samples, FSBPT
    2. Web knowledge sources general Physical Therapy, anatomy, physiology, etc.,
2. Describe the default chunking strategy that you will use. Why did you make this decision?
    1. Used 800 with 100 overlap, for better retrieval of papers' paragraphs content for Qwen embeddings
3. [Optional] Will you need specific data for any other part of your application? If so, explain.

Task 4: Build an end-to-end Agentic RAG application using a production-grade stack and your choice of commercial off-the-shelf model(s)



Task 5: Creating a Golden Test Data Set

⚠ **Due to RAGAS hanging up on CustomNodeFilter process I will work on it again after today**

> **You are an AI Evaluation & Performance Engineer.** The AI Systems Engineer who built the initial RAG system has asked for your help and expertise in creating a "Golden Data Set" for evaluation.
>
> <aside> 📝
>
> Task 5: Generate a synthetic test data set to baseline an initial evaluation with RAGAS
>
> </aside>
>
> ☑ **Deliverables**

1. Assess your pipeline using the RAGAS framework including key metrics faithfulness, response relevance, context precision, and context recall. Provide a table of your output results.
2. What conclusions can you draw about the performance and effectiveness of your pipeline with this information?

> **Task 6: The Benefits of Advanced Retrieval**
>
> **You are an AI Systems Engineer.** The AI Evaluation and Performance Engineer has asked for your help in making stepwise improvements to the application. They heard that “as goes retrieval, so goes generation” and have asked for your expertise.

Task 6: Install an advanced retriever of your choosing in our Agentic RAG application.

⚠ **Due to RAGAS hanging up on CustomNodeFilter process I will work on it again after today**

1. Describe the retrieval techniques that you plan to try and to assess in your application. Write one sentence on why you believe each technique will be useful for your use case.

2. Test a host of advanced retrieval techniques on your application.

Task 7: Assessing Performance

You are the AI Evaluation & Performance Engineer. It's time to assess all options for this product.

Task 7: Assess the performance of the naive agentic RAG application versus the applications with advanced retrieval tooling

⚠ Due to RAGAS hanging up on CustomNodeFilter process I will work on it again after today

☑ **Deliverables**

1. How does the performance compare to your original RAG application? Test the fine-tuned embedding model using the RAGAS frameworks to quantify any improvements. Provide results in a table.
2. Articulate the changes that you expect to make to your app in the second half of the course. How will you improve your application?

Your Final Submission

Please include the following in your final submission:

1. A public (or otherwise shared) link to a **GitHub repo** that contains:
    1. A 5-minute (OR LESS) loom video of a live **demo of your application** that also describes the use case.
    2. A **written document** addressing each deliverable and answering each question
    3. All relevant code