Option 1 : Build a Chat with PDF with evidence/citation as feature for financial statements

Problem Statement:

- A user will upload a PDF file (financial statement of a company) and then try to ask a question
- Assume the question is one sentence, 10-15 words. But may have a descriptive answer, not yes no answer.
- The AI system is able to try to search for answers within the document and respond with the result.
- But along-with the answer, the system also shows a reference , evidence or citation of the information from where the answer is picked up.

Scope / Output / Artifacts to Build

- Assume a single financial statement PDF file but large (> 100) number of pages
- Assume one single question at a time.
- Write the code, short (1 page) explanation of your architecture/system.
- Give a cost estimate of how this system can be hosted to serve ~1000 parallel files. Explain the cost estimate from first principles.

Technology Stack / Choices

- Build it using Python, libraries of your choice.
- Demo using Gradio / simple HTML server / equivalent. Terminal based demo okay if you are crunched on time.
- You *cannot* use any third party API for PDF parsing / extraction (local library okay).
- It is okay to use an LLM API for the query side but you should be able to explain how the system will work in local LLM mode.
- Use in memory data stores , if needed.

Things you need not build in assignment but explain in interview

- How will this work when we get multiple requests, each of large PDF files ?
- What are the main bottlenecks of the system ?
- What are the key choices which are going to affect the accuracy of this system ?
- What are the key design choices which are going to affect the latency of this system ?

- What kind of feedback , experience can we offer to the user to build their trust in the system?
- What happens if the answer to the question lies inside a table ?
- What if the answer to the question lies in a table or paragraphs that spills over to the next page?
- What if the answer to the question is not present in the table ?

Option 2 : A RAG based query solution over images

Problem Statement:

- A user will upload multiple images (put a reasonable restriction on number of images)
- The user will ask if a particular object (imagine a car, a sign-board, a machine) is present in the image.
- The user will describe the object in few words in the query (for example: signage of a retail shop, a black sedan car)
- The system will try to locate it , answer the presence/absence question and also *highlight* the object in the image

Scope / Output / Artifacts to Build

- Assume a mid to high resolution image in png / jpeg format.
- Assume one single object is queried at a time.
- Write the code, short (1 page) explanation of your architecture/system.
- Give a cost estimate of how this system can be hosted to serve ~1000 requests. Explain the cost estimate from first principles.

Technology Stack / Choices

- Build it using Python , libraries of your choice.
- Demo using Gradio / simple HTML server / equivalent. Terminal based demo okay if you are crunched on time.
- It is preferred that you demo this using a locally hosted image model / VLM.
- It is okay to use an LLM API for the query side but you should be able to explain how the system will work in local LLM mode.
- Use in memory data stores , if needed.

Things you need not build in assignment but explain in interview

- How will this work when we get multiple requests, each with few high resolution images ?
- What are the main bottlenecks of the system ?
- What are the key choices which are going to affect the accuracy of this system ?
- What are the key design choices which are going to affect the latency of this system ?
- What kind of feedback , experience can we offer to the user to build their trust in the system?

Code Guidelines

1. Check in your code into a GitHub repository and send us a link. Ideally, we'd like to see the entire history of your check-ins/iterations into Git and we strongly discourage dumping just the final output in Git.
2. Write idiomatic, readable code, use type hints and doc comments.
3. The code should take care of basic formatting, linting, unit testing.
4. The code should be reproducible on another machine, so take care of library versions etc. Preferably use `uv` project for setting up the project. We strongly *discourage* using conda environments for this assignment.
5. For GPUs, you can set up on Modal Labs and use their credits for GPU containers. Or you can use any alternative solution that you can explain in the interview.
6. AI usage : it is okay to use AI assistants. But you should be able to explain all/any functions/methods/classes used in the code.