AI Engineer Coding Challenge

We would like you to do a coding project so we can better understand your technical skills and overall knowledge as an AI engineer. Imagine that other developers will read and modify your code, so they should be able to understand it easily. You should also remember that code should be easily extensible, maintainable, and scalable, but at the same time beware of overly complicated solutions and speculative design

We don't expect a perfect solution or a finished product, the goal is to see that you can design a reasonable solution to a problem and implement a simple version of it. It's up to you how much time you spend on the challenge, but we'd advise you to work no longer than 15 hours on it. You should be able to explain your solution, the design choices, and the implementation. Please make sure to document any assumptions you make while building your solution.

Objective

Your goal is to build a system that lets users input PDF documents and allows them to ask questions related to information contained in those documents. The system should be capable of processing the document, extracting relevant information, and responding to user queries based on the document's content
The system should thus have two input parameters, one to ingest a folder of documents and one to receive a user query. How you design this interface is up to you but is not the primary focus of the project, so a simple CLI is more than enough.

Document Input

- The system should accept a folder of documents.
- The system should be able to ingest a folder containing at most 100 documents, whose total amount of textual data wouldn't fit in a LLM context window.
- The system should accept documents in the PDF formats.
- Documents are expected to be of reasonable size but may contain large amounts of text, and the content could be spread across different sections or pages.
- Documents are expected to be in English.

Query Input

- Users should be able to input questions related to the document's content in English.
- The system should process these queries and extract the most relevant information from the documents to provide accurate answers.
- The system should be able to identify when a query cannot be answered based on the document content and respond appropriately.

Example

I upload a folder containing 10 PDF documents containing non-publicly available information from a financial consulting company (let's call it Consult+). Two of those documents contain information (might be only one part of the file related to this) on Consult+'s prediction of the performance of Tesla stock (fresh data not available anywhere else). Once the folder is ingested, when I ask the query "What is Consult+ prediction for the price of Tesla stock?", I expect the answers to contain the information coming from those two documents.

This is a concrete example to give you a clear sense of what is required, but of course, the query and the information contained in the documents might not be as directly related. Useful information related to the query might be spread among different documents. So the output should contain as much information related to the query as it can get from the documents processed. Ideally, if the system doesn't have any information related to the query, it should let the user know that.

Technology Choices

You are free to choose any technologies, such as special databases, pre-trained language models (LLMs), or text-processing libraries. However, do not use any pre-made tools that would encompass large parts of the project. You should focus on building the solution from the ground up.
While we do not expect a completely flexible and technology-agnostic solution, we do appreciate a design that allows for exchanging certain technologies or services with a reasonable amount of effort.

Documentation

We expect you to include instructions on how to build and run your solution in the README.md file and also include your full name. In case your solution depends on externally hosted services, please specify this.
You can also document any design decisions, technology choices, and other interesting information in the README. Please also document any reasonable assumptions you had to make while building your solution.

Evaluation

It’s an open-ended project and there is no single best solution to it. As stated before, the goal is to see if you can come up with an interesting solution, implement it, and explain your design choices and their implications

Your solution will be evaluated based on these criteria:

- Pertinence of the overall LLM/Data pipeline solution
- Architecture choices
- Extensibility / Separation of Concern
- Code Simplicity & Readability
- Code Structure & Documentation
- Quality/Relevance of the output

Result

When you feel pleased with your solution, please send us an email with a link to a public Git repo or attach the zipped source code of your project.
Your solution will be reviewed internally and in case it passes our evaluation, you’ll be asked to present it during a technical interview and answer questions about the design decisions and trade-offs you made.