Interaction Logs – PDF Summarizer AI Agent

Overview

This document provides an overview of my collaboration with an AI language model (LLM) throughout the development of the PDF Summarizer AI Agent project. The LLM played a key role in assisting with code generation, debugging, and the creation of essential project documentation such as the architecture and data science reports. The collaboration centered on designing a web-based application capable of extracting text from PDF documents using PyMuPDF and generating concise summaries through a fine-tuned language model.

Use of LLM Assistance

Throughout the development process, I leveraged an LLM for multiple aspects of the project. It assisted in generating Python scripts for PDF text extraction, summarization logic, and building the Streamlit-based user interface. The model also helped in diagnosing and resolving dependency-related issues involving PyMuPDF, Streamlit, and relative imports within the src package. Additionally, it contributed to drafting comprehensive documentation, including the README file, architectural overview, and fine-tuning reports. The LLM provided clear step-by-step guidance for environment setup and web application deployment, while also enhancing the clarity and presentation of written deliverables. Overall, the AI model served as an effective technical collaborator, helping me refine both the codebase and supporting documentation to produce a robust and well-structured final project.

Key Prompts Used

During the development process, I frequently collaborated with the LLM to refine the implementation, troubleshoot runtime errors, and ensure smooth deployment of the application. The model assisted me in identifying and resolving technical issues such as incorrect module imports, missing dependencies, and misconfigured project paths.
Through iterative discussions, I was able to understand the underlying causes of these problems and systematically fix them. The LLM also provided detailed guidance for structuring the Streamlit web interface, organizing the codebase, and generating essential project documentation such as the architecture overview and data science report. This collaborative workflow significantly enhanced both the technical quality and my understanding of the end-to-end development process.

Development Flow

The development process began with a detailed examination of the project's codebase, focusing on how the core components — pdf_extractor.py, summarizer.py, and web_app.py — interacted with one another. The LLM guided me through the proper command sequence required to execute the backend summarization module and subsequently launch the Streamlit-based frontend. During testing, I encountered several issues such as import path errors (e.g., *ModuleNotFoundError: No module named 'src'*) and missing dependencies related to PyMuPDF (fitz). With the LLM's assistance, I was able to identify the causes and apply the correct solutions, including reinstalling dependencies and adjusting file references. After achieving a fully functional web interface, the AI model further assisted in preparing technical documentation, summarization notes, and evaluation reports. Finally, the entire project was structured and documented for GitHub deployment, ensuring clarity, usability, and reproducibility.

Reflection

Working alongside the LLM significantly enhanced my development efficiency and provided clarity on intricate implementation details. The model served as both a programming assistant and a writing collaborator, offering valuable insights and generating foundational code when needed. While I maintained full control over testing, debugging, and final decision-making, the AI contributed expert guidance and explanations that streamlined the process. This partnership demonstrates the effectiveness of AI-assisted development in academic and research contexts, enabling faster, more organized engineering workflows while preserving essential human oversight.