Interaction Logs – PDF Summarizer AI Agent

Overview

This document summarizes my interaction and collaboration with an AI language model (LLM) during the development of the **PDF Summarizer AI Agent** project. The AI assistant was used to guide and generate code, debug runtime issues, and prepare supporting documentation such as architecture and data science reports. The interactions focused on building a web-based PDF summarization tool that can extract text from PDF files using **PyMuPDF** and generate summaries via a fine-tuned model.

Use of LLM Assistance

I used an LLM throughout the development cycle for several purposes: Generating Python scripts for PDF text extraction, summarization, and Streamlit-based UI. Debugging dependency errors (PyMuPDF, Streamlit imports, and relative imports for src package). Writing clear documentation such as README, architecture overview, and fine-tuning reports. Providing step-by-step setup commands for local environment configuration and web app execution. Improving textual clarity and formatting in final project deliverables. The AI model acted as a technical collaborator, helping refine code and structure explanations to ensure the final project was fully functional and well-documented.

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 began by analyzing the initial code structure and understanding the interaction between pdf_extractor.py, summarizer.py, and web_app.py. The LLM provided the correct command sequence to execute the backend summarization module and later deploy the Streamlit frontend successfully. During testing, I encountered import path issues (e.g., ModuleNotFoundError: No module named 'src') and missing library errors (fitz / PyMuPDF). These were resolved through guidance from the AI model, which suggested appropriate fixes and confirmed installation steps. Once the web interface was functional, the model helped draft technical documentation, fine-tuning notes, and final evaluation summaries. The project was then organized for GitHub deployment with a clear README, ensuring reproducibility.

Reflection

Collaborating with the LLM accelerated my workflow and clarified complex implementation steps. It acted as both a coding assistant and a writing partner. I retained control of all testing and debugging decisions while the model provided guidance, explanations, and boilerplate generation. This collaboration highlights the practical utility of AI-assisted development in education and research settings — helping streamline the engineering process while maintaining human-driven oversight.