Prompt Log

#	Initial Prompt	Refined Prompt	Reason for Change
1	Generate Python code for a Streamlit app that summarizes meeting transcripts, including imports for Streamlit, Gemini API, VADER sentiment analysis, and PyPDF2 for PDF handling.	Generate Python code for a Streamlit app with imports for Streamlit, google.generativeai, vaderSentiment, PyPDF2, os, and dotenv for secure API key handling. Include a function to load environment variables and configure the Gemini API.	Initial output missed os and dotenv for secure API key management. Refined to ensure proper Gemini API setup after testing showed key loading issues.
2	Write a Python function to extract text from a PDF file using PyPDF2.	Write a Python function extract_text_from_pdf(file) that takes a file object, uses PyPDF2 to extract text from all pages, concatenates with newlines, and strips whitespace. Handle errors if the PDF is invalid.	Initial function processed only one page and lacked error handling. Refined for multi-page support and robustness per PyPDF2 documentation.
3	Generate a Python function to summarize text using the Gemini API.	Write a Python function summarize_key_points(transcript) that uses the Gemini 1.5 Flash model to summarize a meeting transcript into 3-5 bullet points, focusing on main topics, decisions, and outcomes. Format output as Markdown.	Initial output was too generic, producing long summaries. Refined to specify model, limit bullet points, and ensure Markdown formatting after testing.
4	Create a Python function to extract action items from text using Gemini.	Write a Python function extract_action_items(transcript) using Gemini 1.5 Flash to extract action items from a meeting transcript as a Markdown bullet list, including responsible person and deadlines if mentioned.	Initial output missed owner/deadline details and formatting. Refined for structured output based on task extraction research.
5	Write a Python function for sentiment analysis of text.	Write a Python function analyze_sentiment(transcript) using vaderSentiment to analyze text sentiment. Return 'Positive' (compound >= 0.05), 'Negative' (<= -0.05), or 'Neutral' with the compound score formatted to 2 decimal places.	Initial function lacked clear classification thresholds. Refined to match VADER's scoring and improve clarity per documentation.
6	Generate a Streamlit app main function for a text summarizer.	Write a Python main() function for a Streamlit app titled 'Meeting Notes Summarizer'. Include file uploader for PDF/text, text area for pasting transcripts, and a button to process input. Display results in Markdown sections for Key Points, Action Items, and Sentiment. Add a spinner during processing.	Initial UI lacked file upload and structured output. Refined to include all input methods and formatted display per Streamlit tutorials.
7	Combine the above functions into a complete Python script for a meeting summarizer.	Generate a complete Python script meeting_summarizer.py integrating functions for PDF text extraction, Gemini-based key points and action items, VADER	Initial script had integration issues (e.g., no error handling for PDFs). Refined for robustness and security after testing.