

Project Report: AI Assistant Development

Author: Machint Sethi

Position: AI & Prompt Engineering Intern

Date: July 30, 2025

1. Executive Summary

This report details the development of a web-based AI Assistant, a project focused on prompt engineering. The primary objective was to design and implement an AI assistant capable of performing a variety of tasks based on user prompts. The project successfully showcases skills in crafting effective prompts to drive specific AI responses.

The final application is a user-friendly web tool built with Python and Flask. It integrates with the Perplexity AI API and fulfills all core requirements, including offering three distinct functions, varied prompt designs for each function, and a mechanism for users to provide feedback on the AI's performance.

2. Project Requirements

The project was developed based on the specifications outlined in the "Major Project for Prompt Engineering" document. The key requirements are as follows:

- **Functionality:** The AI Assistant must perform at least three distinct functions. The implemented functions are:
 - Answering factual questions.
 - Summarizing a given text.
 - Generating creative content like stories or poems.
- **Prompt Design:** For each function, a minimum of three different prompts were designed to guide the AI's output. These prompts were intentionally varied in their length, specificity, tone, and style to demonstrate effective prompt engineering.
- **User Interaction:** The system required a user-friendly interface for users to input queries, select functions, and view responses clearly. This was achieved through a simple web application.
- **Feedback Loop:** A mechanism for users to provide feedback on the helpfulness of responses was included. The "Was this response helpful? (yes/no)" system logs this data, which can be used to refine prompts and improve AI responses in the future.
- **Documentation:** The project required a brief user guide explaining the assistant's functions and usage.

3. System Architecture and Design

3.1. Technology Stack

- **Backend:** Python with the Flask web framework.
- **Frontend:** HTML, CSS, and vanilla JavaScript.
- **AI Service:** Perplexity API (originally designed for OpenAI).

3.2. Architecture Overview

1. **Frontend (Client):** A single-page web interface (`index.html`) allows users to select a function and prompt style from dropdown menus and enter text into a form. JavaScript dynamically updates the prompt style options based on the chosen function and handles the submission of feedback without reloading the page.
2. **Backend (Server):** The Flask application (`app.py`) receives user requests. It identifies the chosen function and prompt style, constructs the final prompt by inserting the user's text, and sends it to the Perplexity API.
3. **API Integration:** The backend communicates with the Perplexity API, sending the crafted prompt to the `llama-3-sonar-small-32k-online` model. It waits for the response, which is then processed and sent back to the frontend.
4. **Feedback Logging:** When a user submits feedback, the backend receives the request at the `/feedback` endpoint and appends the relevant information—including the timestamp, prompt, response, and user feedback—to a `feedback.log` file.

4. Implementation Details

4.1. Prompt Engineering

A core component of the project was designing varied prompts. This was implemented in `app.py` using a Python dictionary.

Code Snippet: Prompt Design in `app.py`

```
PROMPTS = {
    "question": {
        "Simple": "Answer the following question clearly and concisely: {user_input}",
        "Detailed": "Provide a detailed explanation for the following question...",
        "ELI5": "Explain the answer to this question as if I were 5 years old: {user_input}"
    },
    "summarize": {
        "Key Points": "Summarize the following text into a few key bullet points: {user_input}",
        "Comprehensive": "Provide a comprehensive summary of the following text...",
        "Short Paragraph": "Summarize the following text in a single, well-written paragraph..."
    }
}
```

```
    },  
    # ... and so on for the 'creative' function  
}
```

4.2. API Interaction

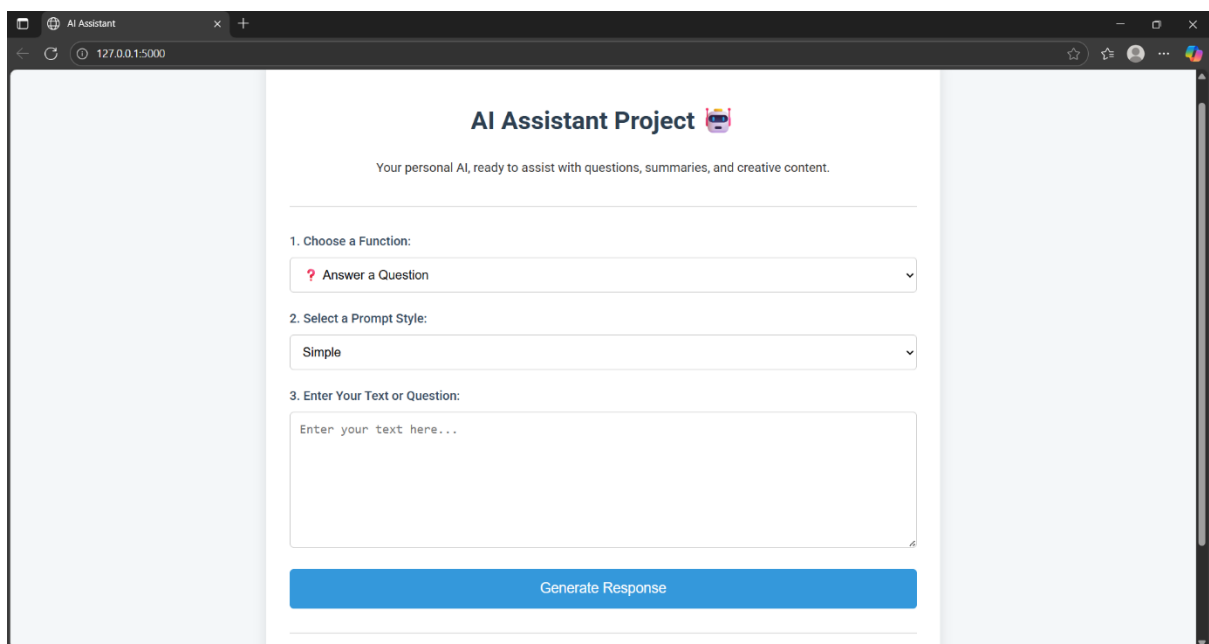
The `get_ai_response` function in `app.py` is responsible for communicating with the external AI service.

Code Snippet: Perplexity API Call

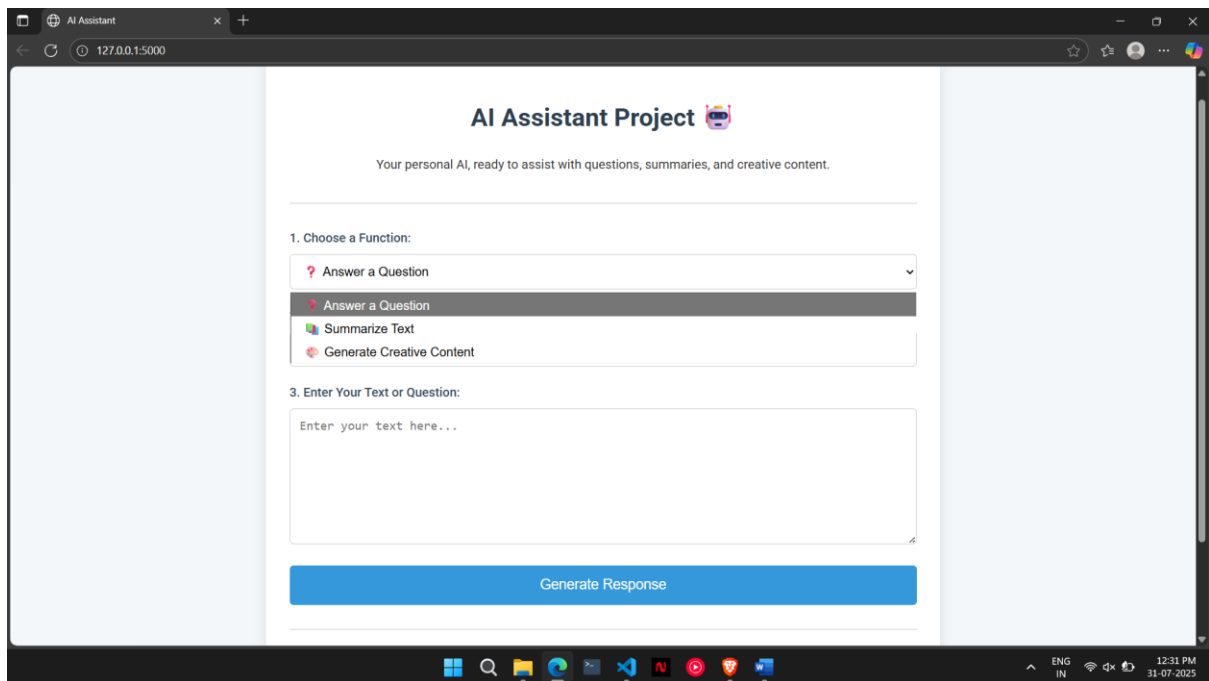
```
def get_ai_response(prompt):  
    try:  
        messages = [  
            {"role": "system", "content": "You are a helpful assistant."},  
            {"role": "user", "content": prompt},  
        ]  
  
        response = client.chat.completions.create(  
            model="llama-3-sonar-small-32k-online",  
            messages=messages,  
        )  
        return response.choices[0].message.content.strip()  
    except Exception as e:  
        return f"An error occurred: {e}"
```

5. Implementation Screenshots

5.1. Home Page



5.2. Choose a Function:



The screenshot shows a web browser window with the title 'AI Assistant' and a URL of '127.0.0.1:5000'. The page is titled 'AI Assistant Project' with a subtitle 'Your personal AI, ready to assist with questions, summaries, and creative content.' Below the title, there is a section '1. Choose a Function:' with a dropdown menu. The dropdown menu is open, showing three options: 'Answer a Question' (selected), 'Summarize Text', and 'Generate Creative Content'. Below the dropdown, there is a section '3. Enter Your Text or Question:' with a text input field containing the placeholder text 'Enter your text here...'. At the bottom of the form, there is a blue button labeled 'Generate Response'. The Windows taskbar is visible at the bottom of the screen, showing the time as 12:31 PM on 31-07-2025.

AI Assistant Project

Your personal AI, ready to assist with questions, summaries, and creative content.

1. Choose a Function:

? Answer a Question

Answer a Question

Summarize Text

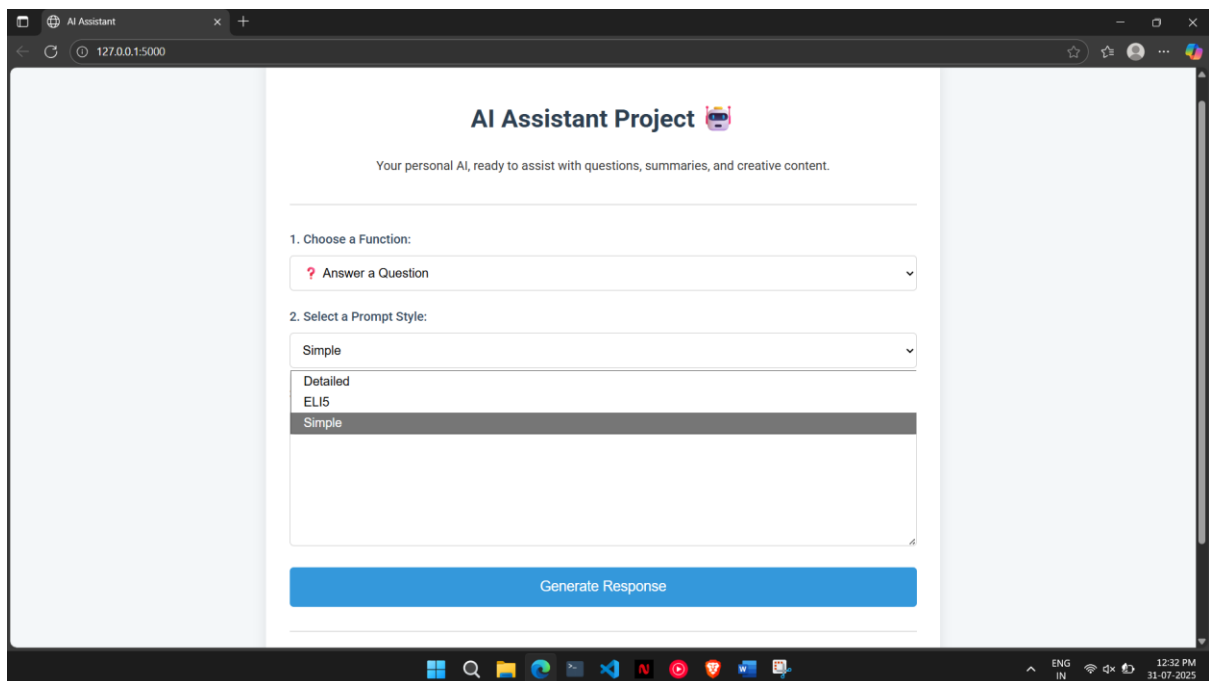
Generate Creative Content

3. Enter Your Text or Question:

Enter your text here...

Generate Response

5.3 Choose the Prompt Style:



The screenshot shows the same web browser window as in the previous image. The page is titled 'AI Assistant Project' with a subtitle 'Your personal AI, ready to assist with questions, summaries, and creative content.' Below the title, there is a section '1. Choose a Function:' with a dropdown menu. The dropdown menu is open, showing three options: 'Answer a Question' (selected), 'Summarize Text', and 'Generate Creative Content'. Below the dropdown, there is a section '2. Select a Prompt Style:' with a dropdown menu. The dropdown menu is open, showing three options: 'Simple' (selected), 'Detailed', and 'ELI5'. Below the dropdown, there is a text input field. At the bottom of the form, there is a blue button labeled 'Generate Response'. The Windows taskbar is visible at the bottom of the screen, showing the time as 12:32 PM on 31-07-2025.

AI Assistant Project

Your personal AI, ready to assist with questions, summaries, and creative content.

1. Choose a Function:

? Answer a Question

2. Select a Prompt Style:

Simple

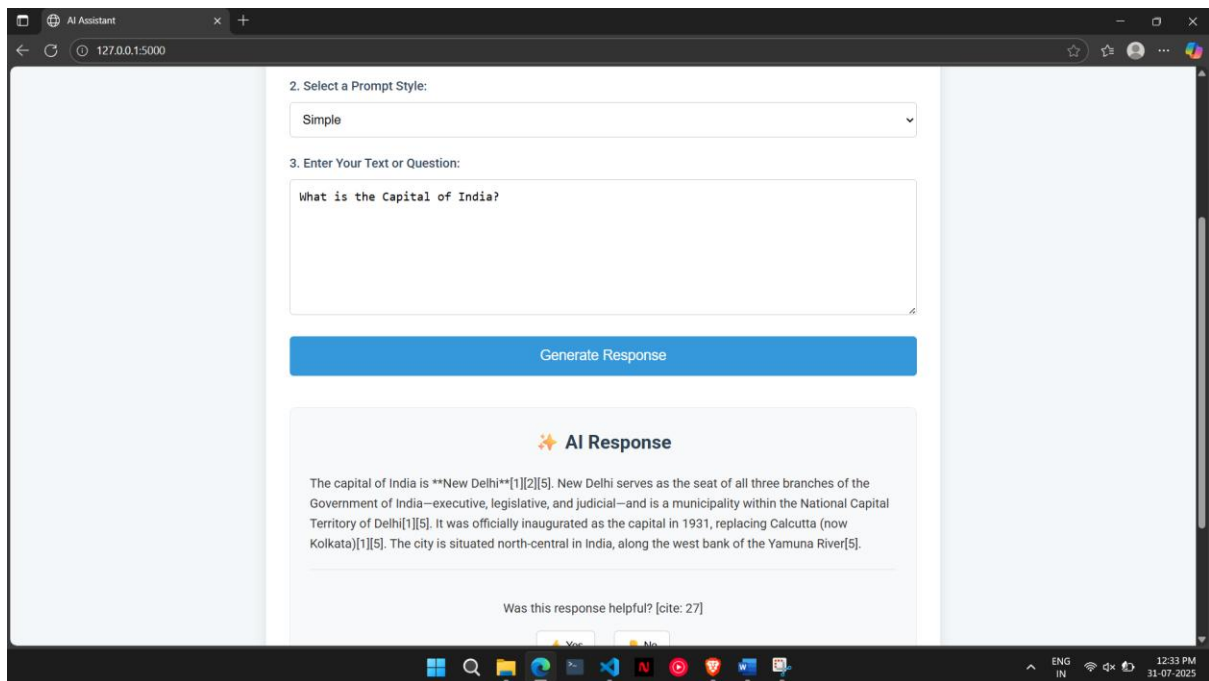
Detailed

ELI5

Simple

Generate Response

5.4. Generate Response:



6. Conclusion and Future Work

6.1. Conclusion

The AI Assistant project was successfully completed, meeting all functional and documentation requirements. It serves as a practical demonstration of prompt engineering principles and provides a solid foundation for building more complex AI-driven applications. The final product is a functional web tool that effectively uses an external AI API to perform diverse tasks based on carefully crafted user prompts.

6.2. Future Work

The current application is a strong proof of concept. The following enhancements could be made in the future:

- **Automated Prompt Refinement:** Analyze the `feedback.log` data to automatically identify and refine underperforming prompts.
- **Expanded Functionality:** Add new functions, such as language translation, code generation, or sentiment analysis.
- **User Personalization:** Implement user accounts to save conversation history and tailor the experience to individual users.
- **Deployment:** Deploy the application to a cloud platform (e.g., Heroku, AWS) to make it publicly accessible.