PROMPTCRAFT AI

Human-in-the-Loop Prompt Engineering

A Final Report Submitted for the Value Added Course

Generative AI & Prompt Engineering

Submitted by

GOPAL KAUSHIK

UID: 25BCS80003

Class: B.E. CSE (LEET)

Department of Computer Science & Engineering

CHANDIGARH UNIVERSITY

Punjab, India

Certificate

This is to certify that the project report entitled **PromptCraft AI** submitted by **Gopal Kaushik** (UID: 25BCS80003) is a bona fide record of work carried out for the Value Added Course **Generative AI & Prompt Engineering** at **Chandigarh University**, Punjab.

Er. Simranjeet Kour
Course Instructor
Dept. of Computer Science
Chandigarh University

Acknowledgement

I would like to express my sincere gratitude to my course instructor, **Er. Simranjeet Kour**, for her guidance and support throughout the duration of this Value Added Course. Her simple and effective explanations of Generative AI concepts helped me understand the importance of good prompting.

I also thank the Department of Computer Science and Engineering, Chandigarh University, for providing the labs and resources needed to build this project.

Gopal Kaushik
UID: 25BCS80003

Abstract

In the world of Artificial Intelligence (AI), the quality of the answer you get depends heavily on the quality of the question you ask. This is often called ”Prompt Engineering.” Many users struggle to get good results from AI because they ask vague or incomplete questions.

To solve this, I built **PromptCraft AI**. It is a tool that acts as a helper between the user and the AI. Instead of sending a bad prompt directly, my tool first ”fixes” the prompt to make it professional and detailed. The user can review this fixed prompt before getting the final answer. This report explains how I built this tool using Python and the Google Gemini API to make AI easier for everyone to use.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Generative AI tools like ChatGPT and Google Gemini are very powerful, but they are not magic. They need clear instructions to work well. This skill—giving clear instructions—is called ”Prompt Engineering.”

1.2 The Problem

Many students and beginners face these common issues:

- **Vague Questions:** Asking ”write code” instead of ”write Python code for a calculator.”
- **Bad Results:** Because the question was vague, the AI gives a generic or wrong answer.
- **Wasted Time:** The user has to keep asking again and again to get the right output.

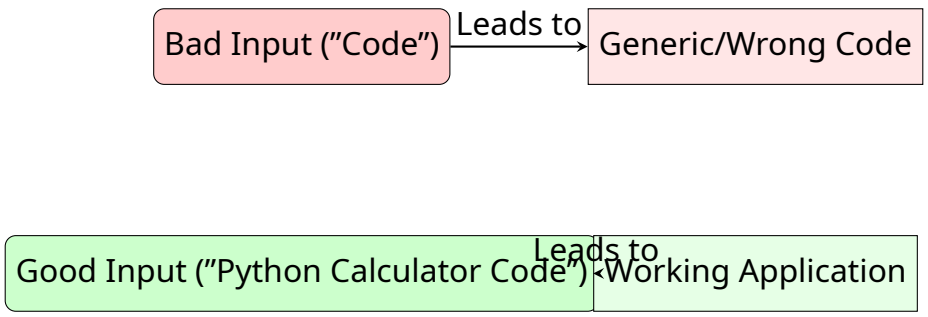

Figure 1.1: The Core Problem: Input Quality Determines Output Quality

1.3 The Solution

My project, PromptCraft AI, solves this by adding a ”Refinement Step.” It automatically improves the user’s rough idea into a perfect prompt.

1.4 Project Objectives

1. To build a tool that understands what the user *wants* to say, even if they don’t say it clearly.
2. To let the user check the improved prompt before using it (Human-in-the-Loop).
3. To create a simple, good-looking interface using Python.

Chapter 2

System Analysis

2.1 Workflow (How the App Runs)

The application follows a simple 3-step process to ensure the best results:

2.1.1 Step 1: Improving the Idea

When a user types a short idea (like ”snake game”), the app doesn’t answer immediately. First, it sends this idea to an ”Expert AI Agent.” This agent rewrites the idea into a detailed paragraph, adding rules, coding languages, and specific requirements.

2.1.2 Step 2: Checking with the User

The app shows this new, detailed prompt to the user. The user can read it and say, ”Yes, this is what I wanted,” or edit it if something is wrong. This creates a feedback loop.

2.1.3 Step 3: Getting the Final Answer

Once the user is happy with the prompt, the app sends it to the main AI (the Executor) to actually write the code or answer the question.

2.2 Workflow Diagram

This diagram shows the path of a user’s request:

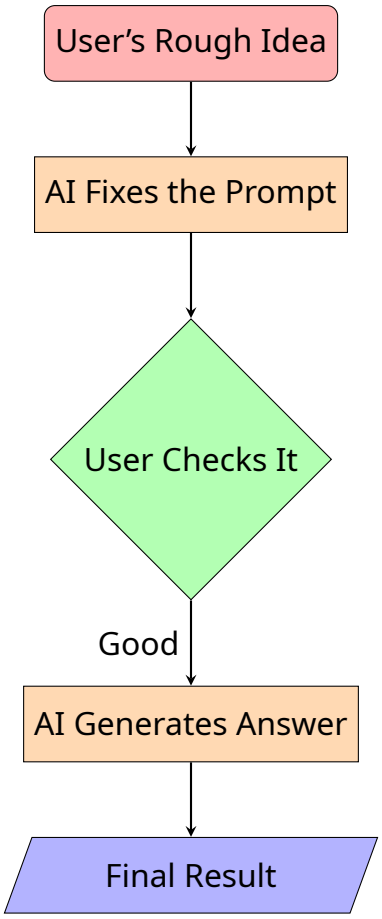

Figure 2.1: Refine-Review-Execute Workflow

Chapter 3

System Design

3.1 Architecture

The project is built using a Client-Server model.

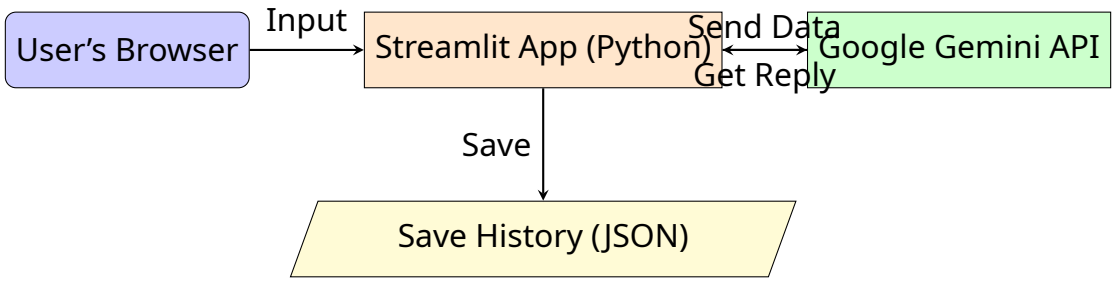

Figure 3.1: System Architecture Diagram

3.2 Data Flow Diagram (DFD Level 0)

This diagram shows how data moves through the system.

Figure 3.2: Data Flow Diagram (Level 0)

3.3 Sequence Diagram (Timeline)

This shows the order of events in time.

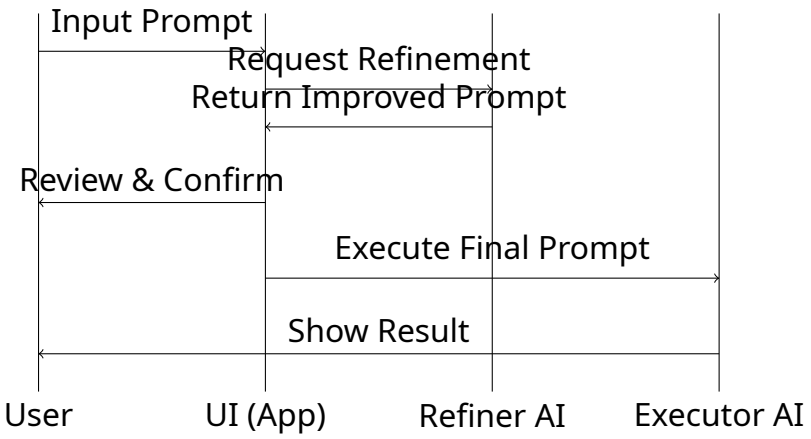

Figure 3.3: Sequence Diagram of Interaction

3.4 Python Files (Modules)

The code is split into three main parts to keep it organized:

- **gemini_client.py (The Brain):** This file handles the connection to Google's AI. It contains the logic to "fix" prompts and "generate" answers.
- **app.py (The Interface):** This file builds the website that the user sees. It handles the buttons, text boxes, and colors.
- **history.py (The Memory):** This file saves the chat history to a file so that previous conversations are not lost.

Chapter 4

Implementation Details

4.1 Tools Used

- **Python:** The main programming language.
- **Streamlit:** A library to create web apps easily using Python.
- **Google Gemini API:** The AI model that powers the application.

4.2 How the ”Fixing” Works

I wrote a special instruction (meta-prompt) that tells the AI to act like an expert. Here is the code logic:

```
def refine_prompt(user_raw_input):
    # This instruction tells the AI how to behave
    instruction = """
    You are an expert Prompt Engineer.
    Rewrite this vague request into a professional prompt.
    User Request: "{user_raw_input}"
    Rules:
    1. Do not answer the question yet.
    2. Just rewrite the question to be better.
    """

    # Send to Google AI
    response = model.generate_content(instruction)
    return response.text
```

Listing 4.1: Logic to Improve Prompts

4.3 Smart Icons (Context Awareness)

The app is smart enough to change the user icon based on what they are talking about. If you talk about code, it shows a ”Coder” icon. If you talk about data, it shows a ”Chart” icon.

```
def get_smart_avatar(text):
    text = text.lower()
    # If user asks about coding
    if "code" in text or "python" in text:
        return " " # Coder Icon
    # If user asks about data
    if "data" in text or "chart" in text:
        return " " # Data Icon

    return " " # Default Robot Icon
```

Listing 4.2: Code for Smart Icons

Chapter 5

Results and Discussion

5.1 Before vs. After

We tested the app with simple inputs to see how much better the outputs became.

User's Short Input	AI's Fixed Prompt	Result
"snake game"	"Write a complete Python script for a Snake game using the Pygame library. Include controls, scoring, and a game over screen."	The AI wrote a working game code because the prompt was specific.
"explain relativity"	"Explain Einstein's Theory of Relativity to a 10-year-old child using simple analogies like trains. No math."	The AI gave a very simple and easy-to-understand explanation.

Table 5.1: Comparison of Raw vs. Improved Prompts

5.2 Performance Comparison

The following chart illustrates the success rate of user queries with and without the refinement step.

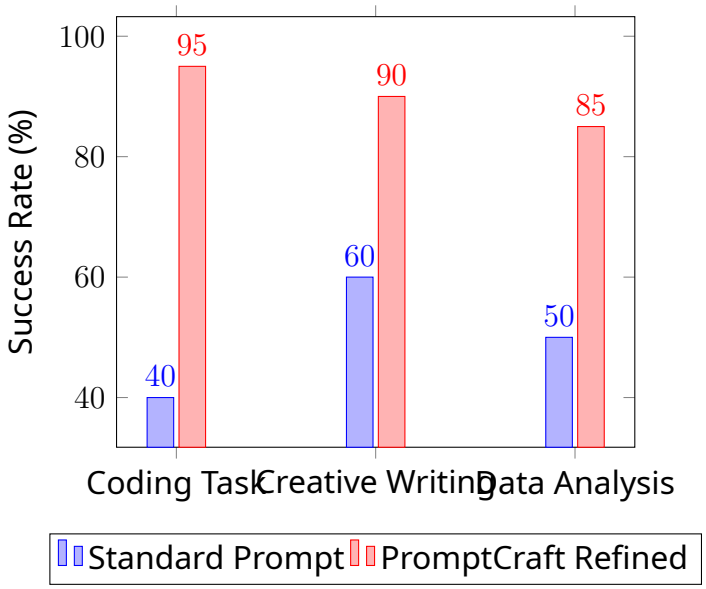

Figure 5.1: Success Rate Comparison: Standard vs. Refined Prompts

Chapter 6

Conclusion

6.1 Conclusion

PromptCraft AI shows that ”Human-in-the-Loop” is a great way to use AI. By letting the AI fix our questions before we ask them, we get much better answers. This project helped me learn how to use APIs, how to build web interfaces with Streamlit, and most importantly, how to communicate effectively with AI models.

6.2 Future Scope

In the future, I plan to add:

- **Voice Support:** So users can speak their ideas instead of typing.
- **Image Generation:** To help users write prompts for creating images.
- **Save Favorites:** A feature to save the best prompts for later use.

References

1. Google AI for Developers. (2024). *Gemini API Documentation*. Available at: `https://ai.google.dev`.
2. Streamlit. (2024). *Streamlit Documentation: Building Data Apps*. Available at: `https://docs.streamlit.io`.
3. OpenAI. (2023). *Best Practices for Prompt Engineering*.