# Poetic Personality Chatbot — Project Report

Introduction

This project builds a chatbot that transforms every user message into short, poetic verses. The chatbot uses a large language model (Gemini) guided by a purpose-built prompt so each response is lyrical, emotionally attuned, and concise. A typing animation effect simulates a live poet composing in real time.

Steps

1. API Setup: Install and initialize the Google Gemini SDK and configure an API key (in Colab, store this in userdata secrets).

2. Prompt Engineering: Create a system instruction that sets tone, length (2–6 lines), stylistic constraints (avoid clichés), and handling of advice requests.

3. Chat Flow: Build a function that sends the user input plus the system prompt to the model and returns the generated text.

4. Typing Effect: Implement a streaming or word-by-word print helper to create the illusion of a poet composing the reply in real time.

5. Packaging: Clean the notebook, add comments, and provide a short project report for submission.

Challenges

**1. Prompt not matching the requirement**

* **Problem:** The original prompt instructed the model to respond with *funny jokes*, which did not match the “poetic personality” requirement.
* **Solution:** Replaced the prompt with a carefully crafted poetic system instruction that defines tone, style, and structure, along with a few-shot example.

2. **Prompt variable bug**

* **Problem:** In the ask\_gemini function, "(user\_input)" was used as a literal string instead of embedding the actual user\_input variable, meaning the model never received user messages.
* **Solution:** Updated the code to properly insert user\_input into the prompt so responses are relevant to the user’s text.

3. **Duplicate stream function**

* **Problem:** The stream function was defined twice in the notebook, causing redundancy and potential confusion.
* **Solution:** Removed the duplicate definition and kept a single, clearly commented version.

4. **Empty code cell**

* **Problem:** One cell in the notebook was blank, making the project look unfinished.
* **Solution:** Removed the empty cell and ensured all code cells have functional content.

5. **Minimal comments**

* **Problem:** The notebook lacked step-by-step explanations, making it hard for readers to understand the workflow.
* **Solution:** Added clear markdown cells and inline comments explaining API setup, prompt crafting, chatbot logic, and streaming animation.

6. **Lack of error handling for missing API key**

* **Problem:** If the API key was missing, the chatbot would fail silently.
* **Solution:** Added exception handling to detect missing keys and show a clear error message.

7. **Response tone drift**

* **Problem:** Without strong poetic constraints, the model sometimes produced generic or unrelated text.
* **Solution:** Strengthened the system prompt with style constraints, examples, and guidance to keep output lyrical and on-topic.

Conclusion

The resulting chatbot provides an engaging, poetic interface for users to receive reflective verses tailored to their messages. It is a lightweight demo suitable for Colab and can be extended with a web UI and better streaming for production.

Reflection (approx. 200 words):  
Prompt engineering is the core creative technique in shaping an LLM's output. By carefully framing the system instruction — specifying tone, length, and example responses — we can reliably steer the model toward generating lyrical, emotionally resonant verses rather than generic or humorous replies. Few-shot examples demonstrate the desired structure and serve as anchors for style. The balance between creativity and constraint is delicate: overly tight constraints can stifle poetic language, while too much freedom leads to drift. Iterative testing with edge-case prompts (requests for advice, expressions of grief, or neutral statements) helps refine the prompt and identify when safety filters or content guidance are necessary. Finally, small UX details — like a typing animation — significantly improve the user experience, making the bot feel alive. Overall, prompt engineering turns an opaque model into a dependable creative partner.