Roadmap del Proyecto de Feedback con LLM – 7 Fases

Fase 0 – Setup & Alcance

• Definir MVP: entrada = texto de feedback, salida = respuesta generada por LLM.
• Redactar historias de usuario básicas: «Como usuario quiero…».
• Crear repositorio Git y estructura de carpetas (backend, frontend, README…).
• Configurar entorno virtual e instalar dependencias esenciales: flask, streamlit, groq.
• Añadir archivo .env con GROQ_API_KEY y otras variables.

Fase 1 – Backend Flask – Ruta /feedback

• Implementar endpoint POST /feedback que reciba JSON {"texto": "..."}.
• Responder provisionalmente con «Recibido: ».
• Probar localmente con curl o Insomnia.
• Agregar manejo de errores básicos (texto vacío = 400).

Fase 2 – Integrar LLM con Groq

• Instalar cliente Python de Groq: pip install groq.
• Crear función responder_llm(texto) que llame a la API con prompt adecuado.
• Sustituir respuesta fija por la generada por responder_llm.
• Capturar excepciones de la API y responder 503 en caso de fallo.

Fase 3 – Persistencia con SQLite

• Crear base de datos feedback.db y tabla feedback(id, texto, respuesta, fecha).
• Añadir helper guardar(texto, respuesta) para insertar registros.
• Crear endpoint GET /history que devuelva los últimos N registros en JSON.

Fase 4 – Frontend Streamlit

• Construir interfaz sencilla con st.text_area y botón Enviar que llame a /feedback.
• Mostrar la respuesta del asistente y, opcionalmente, el histórico (/history).
• Añadir st.spinner("Generando respuesta…") para mejorar UX.

Fase 5 – Prompt Engineering

• Separar system prompt y plantilla de usuario en variables.
• Experimentar con temperature, max_tokens y few-shot examples.
• Registrar resultados en CSV para iterar sobre la calidad de respuesta.

Fase 6 – Moderación & Validación

• Implementar lista de palabras prohibidas y rechazar feedback ofensivo (400).
• Limitar longitud del texto y establecer timeout de la llamada al LLM.
• Prevenir coste excesivo rechazando entradas repetidas o vacías.

Fase 7 – Deploy & CI/CD

• Crear Dockerfile con backend y frontend en un contenedor.
• Desplegar en Render/Fly.io/Heroku configurando variables de entorno.
• Configurar GitHub Actions para build y despliegue automático en cada push.