Resumen del Proyecto

Objetivo: Procesar consultas externas, categorizarlas, buscar documentos relevantes y generar respuestas usando un LLM (como GPT o Claude).
Stack sugerido:

Backend: Python + LangChain o LlamaIndex

Frontend: Streamlit / React

LLM: OpenAI (GPT-4o), Claude, o Amazon Bedrock

Vector DB: FAISS (local) o Pinecone (cloud)

Infraestructura: AWS (Lambda, S3, API Gateway, IAM)

Fases del Proyecto

▧ FASE 1 – Diseño y Setup Inicial
Diseño conceptual (listo)

Solicitá recursos en Jira (criterios 2, 3 y 4):

Nuevo repo GitHub

Acceso a deploy AWS (Lambda, S3, API Gateway)

Dependencias necesarias (APIs, SDKs, claves, embeddings)

Estructura del repositorio

bash
Copiar
Editar

```
llm-query-flow/
├── app/
│   ├── main.py # endpoint o interfaz Streamlit
│   ├── categorize.py # categorización
│   ├── search_docs.py # búsqueda en base de documentos
│   └── generate_answer.py # integración con el LLM
├── data/ # documentos cargados
├── vector_store/ # FAISS u otro motor
└── requirements.txt
```

▨ FASE 2 – Desarrollo del Backend

Carga y preprocesamiento de documentos

Extraer texto, dividir en chunks, limpiar y almacenar en data/.

Usar embeddings (OpenAIEmbeddings, HuggingFaceEmbeddings, etc.)

Guardar índices en FAISS.

Categorización

Implementar modelo de clasificación (modelo entrenado o reglas).

Ej: Prompt + LLM → "¿Esta pregunta se refiere a materiales, logística, precios...?"

Búsqueda semántica

Usar embeddings + FAISS para recuperar los documentos más similares.

Fallback si no hay documentos relevantes: usar solo LLM.

Generación de respuesta

Armar prompt del tipo:

txt
Copiar
Editar
Contexto: [Fragmento1] [Fragmento2] [Fragmento3]
Pregunta: ¿Qué tipo de cemento se recomienda para techos?
Respuesta:
Enviarlo al LLM y mostrar resultado.

▨ FASE 3 – Frontend y Demo

Autenticación por whitelist

Validar que el usuario (Customer A o B) esté habilitado.

Interfaz (Streamlit o React)

Campo de texto para pregunta

Filtros: categoría y subcategoría (opcional)

Respuesta generada

Visualización de los documentos usados (transparencia)

Dashboard demo

URL: softwarefactoryai.com/chat/demo

Mostrar categorías + interacción con preguntas reales

▧ FASE 4 – Testing y Validación

Implementar y probar los 5 escenarios de prueba:

Pregunta relevante

Sin documentos relevantes

Fuera de ámbito

Pregunta ambigua

Pregunta que requiere detalle exacto

Checklist de criterios de aceptación:

Validar que cada escenario tenga lógica, fallback y respuesta adecuada.

▧ FASE 5 – Deploy en AWS

Infraestructura mínima en AWS:

API Gateway + Lambda (si es backend puro)

S3 (para los documentos y configuración)

IAM para permisos

(opcional) ECS o EC2 si usás FAISS con gran volumen

CI/CD con GitHub Actions

Testeo + deploy automático al hacer push en main