Customer Support copilot

Context

At Atlan, our customer support team is the backbone of our customer relationships. They handle a wide range of support tickets daily - from simple “how-to” questions to complex technical bug reports.

As we scale, efficiently managing these tickets is crucial to ensuring our customers get the timely and accurate help they need.

An AI Copilot could act as a powerful assistant for our team by automating triage and drafting intelligent responses.

Problem statement

Your challenge is to build the core AI pipeline that can ingest, classify, and respond to customer support tickets.

To showcase your work, you’ll present this pipeline through a functional “dummy helpdesk” application. The primary focus is the AI pipeline; the helpdesk serves as the demo interface.

Core features

Your helpdesk application must demonstrate two primary functionalities:

1. **Bulk ticket classification dashboard:**
    a. On load, the application should ingest the tickets from the provided sample_tickets file.
    b. The UI must display each ticket alongside the detailed classification generated by your AI pipeline, following this schema:
        i. **Topic Tags:** Use relevant tags like How-to, Product, Connector, Lineage, API/SDK, SSO, Glossary, Best practices, and Sensitive data.
        ii. **Sentiment:** e.g., Frustrated, Curious, Angry, Neutral.
        iii. **Priority:** e.g., P0 (High), P1 (Medium), P2 (Low).
2. **Interactive AI agent:**
    a. The UI must provide a text input field where an evaluator can submit a new ticket or question
    b. At Atlan, a customer can reach out via various channels (e.g., WhatsApp, email, voice, live chat).
    c. When a new query is submitted, your UI should clearly display both the **AI's internal analysis (the "back-end" view)** and the **final response (the "front-end" view)**.

i. **Internal Analysis View:** Show the classification details for the new ticket (Topic, Sentiment, Priority).
ii. **Final Response View:**
    1. If the topic is How-to, Product, Best practices, API/SDK, or SSO, the agent must use a RAG pipeline to generate and display a direct answer.
    2. If the topic is anything else, the agent should display a simple message indicating the ticket has been classified and routed (e.g., "This ticket has been classified as a 'Connector' issue and routed to the appropriate team.").

d. **Knowledge base for RAG:**
    i. To answer product-related questions, refer to Atlan's Documentation: https://docs.atlan.com/
    ii. To answer API/SDK related questions, refer to the Developer Hub: https://developer.atlan.com/
e. **Output:** All RAG-generated answers must cite the sources (list of URLs) used to create them.

AI & Engineering requirements

1. **AI logic:** Clean, well-structured code for the classification and RAG pipelines. Your choice of models and tools should be deliberate and explained.
2. **Application:** The helpdesk should be built using a tool like Streamlit, Gradio, Replit, or a simple web framework (e.g., Flask/React) to serve as the interface for your AI.
    a. Use these tool as directional / use any tooling of your choice to solve the problem
3. **Execution:** The entire experience should be smooth and functional in the deployed application.

Deliverables

1. **Working helpdesk application:**
    a. A link to your live, deployed application on a platform like Vercel, Replit, Railway, Streamlit Community or any other cloud platform. This is the primary submission item for evaluating your AI.
2. **Code repository:**
    a. A link to a GitHub repository containing all your code and necessary files.
3. **Documentation (README.md):**
    a. A detailed README.md file in your repository covering:
        i. Major design decisions and trade-offs for your AI pipeline.
        ii. An **architecture diagram** (embedded in the README or linked) illustrating how the components of your system interact.
        iii. Clear instructions on how to set up the environment and run your project locally.

Evaluation criteria

1. **Implementation of core features:** The successful execution of the classification dashboard and the interactive agent, along with any creative enhancements.
2. **Problem-Solving & Approach:** Logical design of the AI pipeline and an effective UI that clearly demonstrates the pipeline's capabilities.
3. **Accuracy of answers:** How do we measure the quality of the answers generated by the model?
4. **Maintainability & engineering decisions:** Quality of the AI pipelines, thoughtful model/tool choices, system architecture, and clean, well-structured code.
5. **Communication:** Clarity of explanation in your documentation

This challenge is your chance to design an AI copilot that could genuinely make customer support more efficient at scale.

We're excited to see how you combine classification, RAG, and thoughtful design to create something impactful! ✨