Principal Software Engineering Contact-Center Mini AI Orchestrator

Objective

Design a **scalable** and **reliable** service that uses a minimal AI classifier (single LLM call) that classifies incoming customer messages from different channels (e.g. chat, voice, mail) into the 3 following categories:

- **informational category,** for example "*What is your refund policy for prescription products?*"
- **service_action** category**,** for example: "*I need to open a ticket because my order never arrived.*"
- **safety_compliance** category**, f**or example: "*I experienced a severe headache and nausea right after taking the medication.*"

Minimum Requirements

- Specs: Docker container, MacOS, or Ubuntu; Python version >=3.10; package management, testing framework, and pre-commit routines of your choice.
- Architecture design to support a scalable production-ready solution that can also support unexpected spikes of demand.
- Interface: A FastAPI service with several endpoints.
- AI Classifier (minimal)
    - Invoke one AI model call that attempts to classify the message into the three categories.
    - You define prompts, confidence scale, and acceptance criteria.
- Single endpoint that accepts a short message payload and returns a structured response
    - Category
    - Confidence score
    - Decision path
    - Next step
- Design E2E workflows for each category separately and provide a lightweight implementation for each of them up until the point where integrations with external systems might be needed. As for the channels, you should design the system to support different channels, but for the implementation part you can use only the simple chat message.
- Build a continuous integration and continuous deployment pipeline (CI/CD).

Deliverables

1. Source Code Repository with clear instructions
2. A single, concise document (or a small set if you prefer) that includes:
    - System architecture overview with diagrams and flows along with the design rationale
    - Assumptions & tradeoffs made
    - Code testing approach
    - Evaluation method and sample results
    - Online monitoring approach
    - CI/CD pipeline description

Evaluation Criteria

- Scalability (system design)
- Code Reusability/Maintainability
- System Reliability (tests & resilience)
- Compliance (handle sensitive information)
- Evaluation (assess/improve quality)
- Online monitoring

Notes

- For the AI classifier part, you are advised to use ollama OSS models, which aren't meant for production environments. You may use proprietary models via APIs, but you will need your own subscription key for this.
- Coding assistants are permitted, but you must be able to answer questions about the code and the decisions made.