Support Ticket Triage Copilot

Product & Technical Specification Document (Client / Product Owner)

1. Introduction

This document defines the functional, technical, and operational requirements for the Support Ticket Triage Copilot. The objective is to build a production-ready machine learning system capable of classifying, prioritizing, and extracting structured information from customer support tickets, following industry-grade MLOps practices.

2. Business Objectives

- Reduce customer support response time.
- Automate ticket categorization and priority assignment.
- Extract key technical and contextual information.
- Provide a structured and scalable ML inference service.

3. Scope of Delivery

- Text classification model (category + priority).
- Information extraction subsystem.
- REST API for inference.
- Training, evaluation, and deployment pipelines.
- CI/CD and reproducibility tooling.

4. Functional Requirements

4.1 Ticket Classification

- Input: Ticket subject and body text.
- Output: Category ∈ {billing, bug, feature_request, account_access, security, other}.
- Output: Priority ∈ {P0, P1, P2, P3}.

4.2 Information Extraction

- Product name (string or null).
- Affected module (string or null).
- Environment ∈ {prod, staging, dev, unknown}.
- Error codes (list of strings).

5. Non-Functional Requirements

- CPU-only inference.
- Inference latency < 250 ms per request.
- Stateless API design.
- Deterministic and reproducible training.

6. Data Requirements

- Open-source or synthetically generated dataset.

- No private or proprietary data.
- Dataset versioned via DVC.
- Data Card documenting source, schema, and limitations.

7. Model & Training Strategy

- Baseline: TF-IDF + Linear model.
- Final model: Transformer-based encoder (DistilBERT or equivalent).
- Multi-head classification for category and priority.
- Sequence labeling or hybrid rule-based extraction.

8. Evaluation Criteria

- Category classification Macro F1 ≥ 0.80.
- Priority classification Macro F1 ≥ 0.70.
- Extraction metrics clearly defined and reported.
- Confusion matrices and error analysis.

9. Deployment & MLOps Requirements

- FastAPI-based REST service.
- Dockerized inference service.
- Health check endpoint.
- Structured logging and basic metrics.

10. CI/CD Requirements

- Automated linting and formatting.
- Unit and integration tests.
- Model regression checks.

11. Final Deliverables

- GitHub repository with tagged release v1.0.
- README with setup, training, and usage instructions.
- Model Card and Data Card.
- Running Dockerized API.

12. Acceptance Criteria

The solution will be accepted if all functional requirements are met, evaluation thresholds are satisfied, the system is reproducible end-to-end, and the API can be deployed and tested locally using Docker without external dependencies.