Capstone Project Proposal: LLM Egress Guard

Using Policy-Driven Detection, Lightweight ML, and a Simple /guard API

Team member: Baran Akin (Student ID: 42546078)

Project Overview

Modern enterprises are rapidly adopting LLMs in apps, agents, and RAG systems but they risk leaking secrets, PII, dangerous commands, and risky URLs in model responses. LLM Egress Guard is a model-agnostic, low-latency service that inspects LLM outputs and masks, delinks, blocks, or annotates risky content using a policy-driven pipeline with lightweight ML to reduce false positives. The MVP runs on a single VM with Docker Compose and Nginx (no streaming), exposing a simple POST /guard API that any LLM application can call.

Problem Statement

1. LLM responses can inadvertently contain: Secrets (API keys, JWTs), PII, shell/PowerShell one-liners, and risky URLs (IP links, data URLs, executable downloads).
2. This leads to: Data exposure, compliance risk, and operational compromise (e.g., copy-pasted curl|bash commands).
3. Need: A deterministic, low-latency egress guardrail with clear policies, telemetry, and minimal privacy footprint decoupled from the underlying LLM/provider.

Objectives

- Detect & Neutralize Risky Content: Apply policy-controlled actions mask, delink, block, annotate on LLM outputs.
- Low Latency: Target p95 < ~80 ms for 1–3K-char responses on a single vCPU (student target).
- Lightweight ML: Integrate a small pre-classifier to suppress false positives on code/command vs. narrative text; optional NER validator later.
- Simple Integration: Provide a minimal /guard API plus health and metrics endpoints.
- Privacy by Design: No raw sensitive text in logs; only masked snippets and snippet hashes.

MVP Scope

- Pipeline: normalize → parse → detect → decide → act → log
- Detectors: PII, Secrets, URL Risk, Command
- Policy: YAML tiers, allow/deny lists, rule IDs & actions

- Runtime: VM + Docker Compose + Nginx; no streaming in MVP
- Telemetry: p50/p95 latency, rule-hit counters
- Tests: ≥100 labeled samples + regression runner

Data Description

Primary (project-generated)

- Synthetic corpus of LLM responses with labeled risky patterns (emails/phones/IBANs, JWT/API keys, IP/data URLs, curl|bash signatures).
- Regression corpus (golden_v1.jsonl) for repeatable evaluations across versions.

Secondary (optional/auxiliary)

- Public regex heuristics and pattern lists for secrets/PII variants; language-specific PII packs (TR/EN/DE) planned post-MVP.

System Architecture

MVP (no streaming): Client → HTTPS 443 → Nginx (TLS, rate limit) → FastAPI /guard
Inside the app:
1) Normalize (Unicode NFKC, ZWSP removal, bounded unescape)
2) (Optional) ML Pre-Classifier (code/command vs. text)
3) Detect (regex/entropy/denylist across PII, Secrets, URL, CMD)
4) (Optional) NER Validator for ambiguous PII hits
5) Decide (policy tiers, allowlist)
6) Act (mask/delink/block/annotate)
7) Log + Metrics (masked snippets + hash; Prometheus at /metrics)

Deployment: Single VM, Dockerized app + Nginx TLS termination, health checks, restart policy.

Expected Output

- Sanitized Responses: Risky spans masked/delinked, or responses blocked with user-friendly safe messages.
- Actionable Findings: Structured response listing rule_id, action, offsets, and optional risk score.
- Telemetry: p50/p95 latency and rule-hit counters for demo and tuning.
- Demo Scenarios: (1) PII mask (e.g., email/phone in an LLM answer), (2) JWT/API key block, (3) curl|bash block + safe guidance.

Tools and Technologies

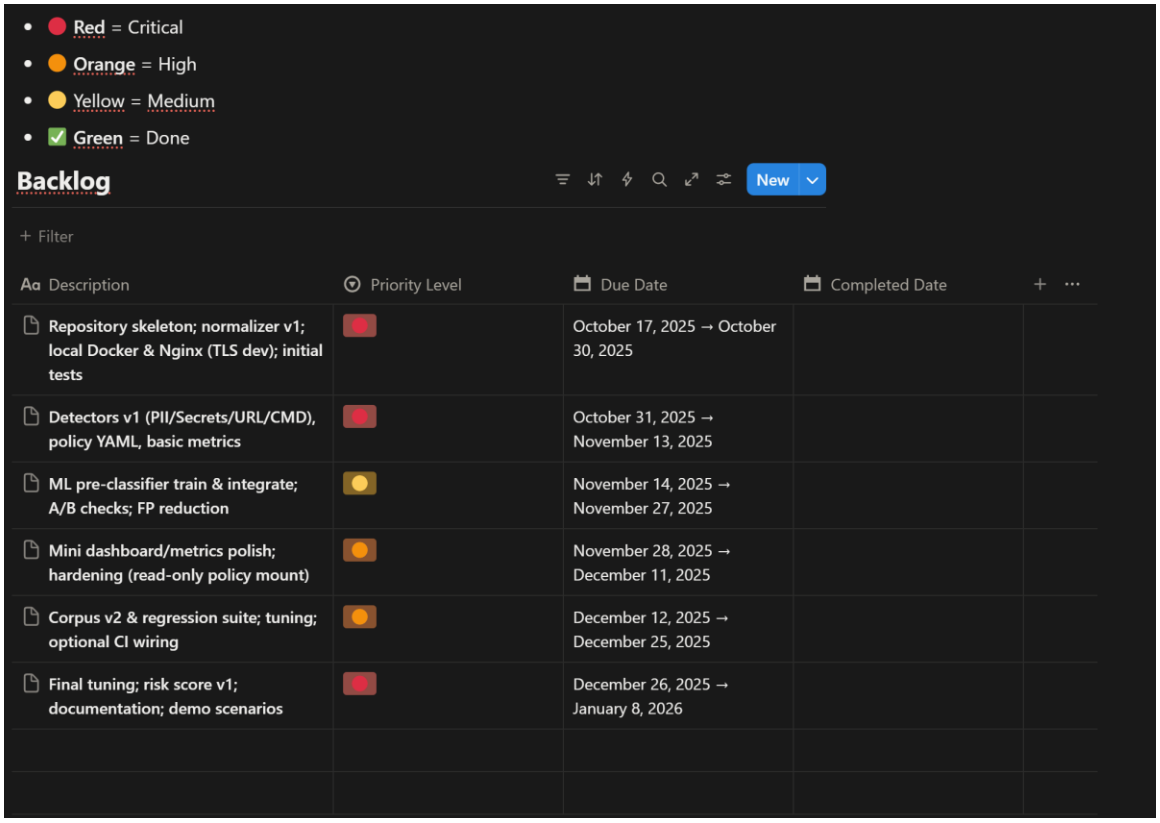
- App & API: Python, FastAPI, Docker Compose, Nginx
- Detection: Regex/entropy/heuristics; modular detectors (PII/Secrets/URL/CMD)
- ML (lightweight): TF-IDF + Logistic Regression pre-classifier; optional spaCy NER validator
- Observability: Prometheus-style metrics endpoint
- Testing: Unit tests + regression runner with golden outputs

Implementation Plan

(Structure adapted to the “Urban Mobility Optimization” proposal format phased plan with dates. Images/Gantt to be added later.)

Phase	Task	Duration	Dates
1	Repository skeleton; normalizer v1; local Docker & Nginx (TLS dev); initial tests	2 weeks	17.10.25 – 30.10.25
2	Detectors v1 (PII/Secrets/URL/CMD), policy YAML, basic metrics	2 weeks	31.10.25 – 13.11.25
3	ML pre-classifier train & integrate; A/B checks; FP reduction	2 weeks	14.11.25 – 27.11.25
4	Mini dashboard/metrics polish; hardening (read-only policy mount)	2 weeks	28.11.25 – 11.12.25
5	Corpus v2 & regression suite; tuning; optional CI wiring	2 weeks	12.12.25 – 25.12.25
6	Final tuning; risk score v1; documentation; demo scenarios	2 weeks	26.12.25 – 08.01.26

Backlog

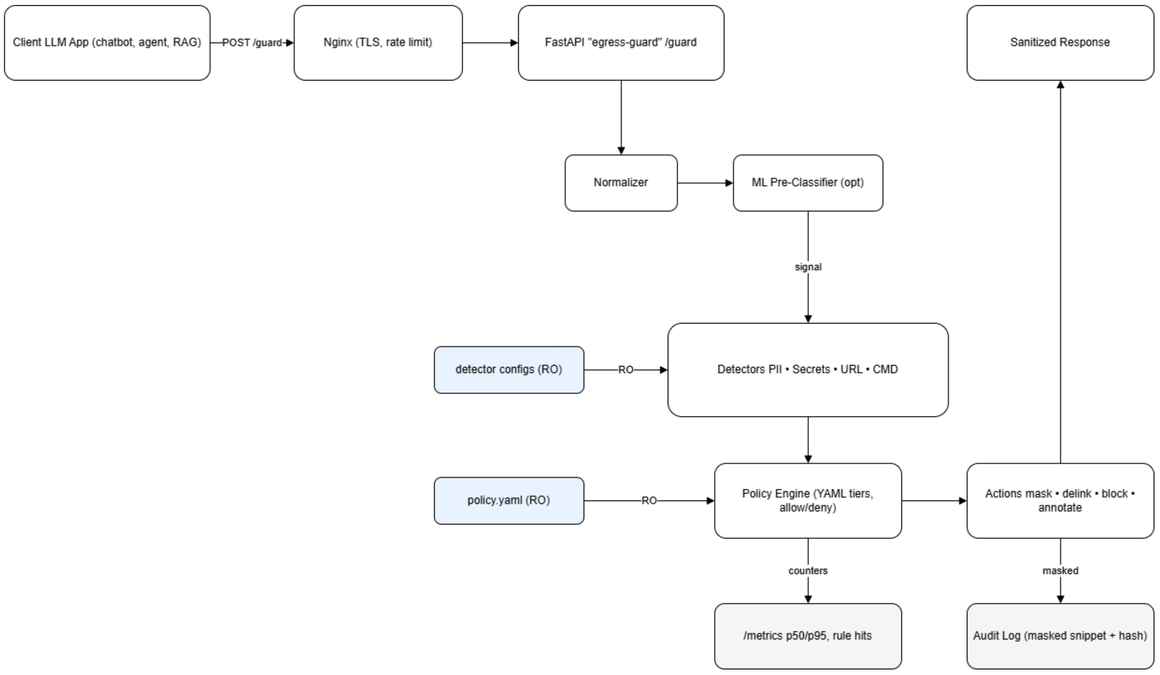


- 🔴 Red = Critical
- 🟠 Orange = High
- 🟡 Yellow = Medium
- ✅ Green = Done

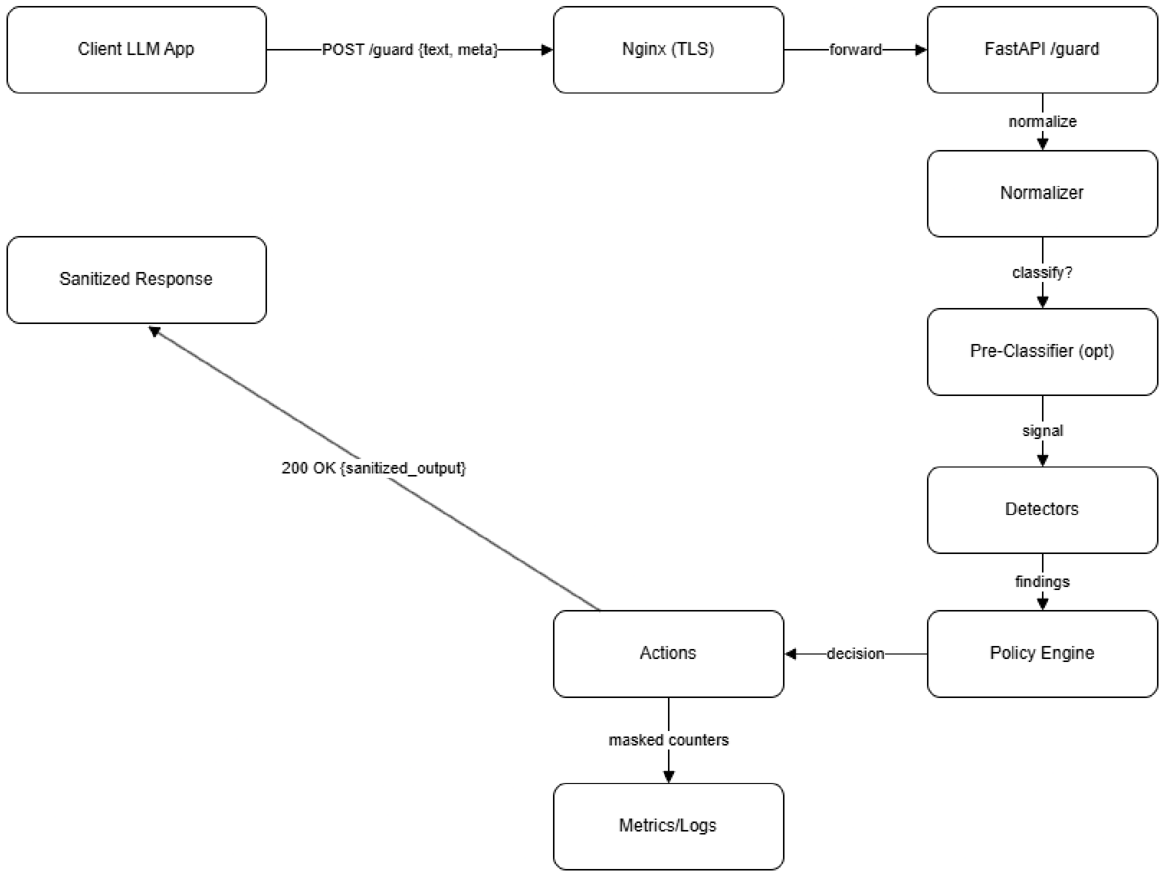
Backlog

Description	Priority Level	Due Date	Completed Date
Repository skeleton; normalizer v1; local Docker & Nginx (TLS dev); initial tests	🔴	October 17, 2025 → October 30, 2025	
Detectors v1 (PII/Secrets/URL/CMD), policy YAML, basic metrics	🔴	October 31, 2025 → November 13, 2025	
ML pre-classifier train & integrate; A/B checks; FP reduction	🟡	November 14, 2025 → November 27, 2025	
Mini dashboard/metrics polish; hardening (read-only policy mount)	🟠	November 28, 2025 → December 11, 2025	
Corpus v2 & regression suite; tuning; optional CI wiring	🟠	December 12, 2025 → December 25, 2025	
Final tuning; risk score v1; documentation; demo scenarios	🔴	December 26, 2025 → January 8, 2026	

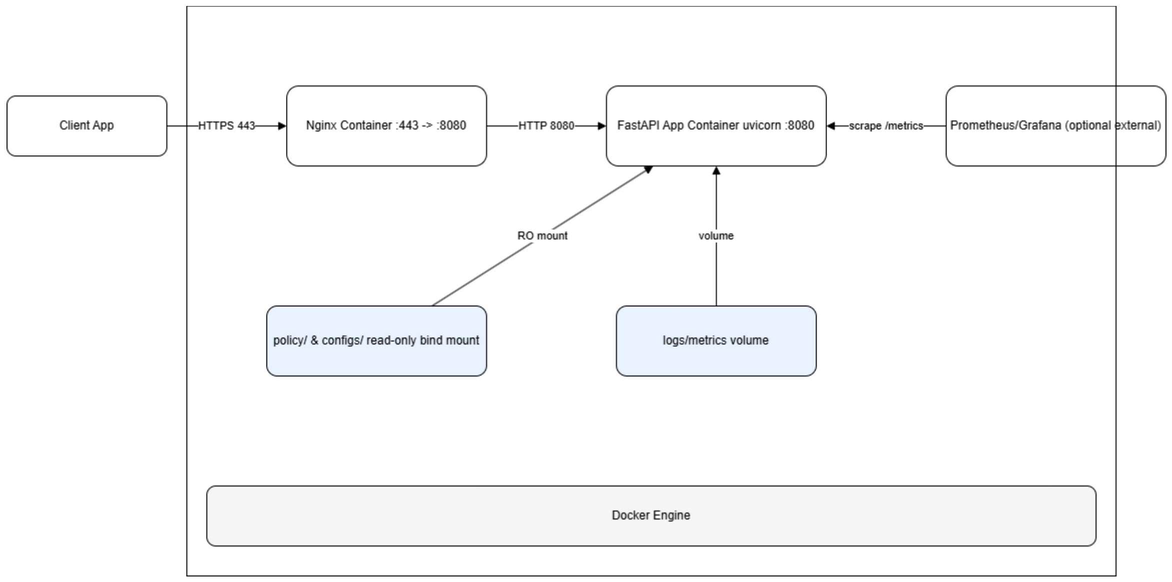
System Design


Client LLM App (chatbot, agent, RAG)
POST /guard
Nginx (TLS, rate limit)
FastAPI "egress-guard" /guard
Sanitized Response
Normalizer
ML Pre-Classifier (opt)
signal
detector configs (RO)
RO
Detectors PII • Secrets • URL • CMD
policy.yaml (RO)
RO
Policy Engine (YAML tiers, allow/deny)
Actions mask • delink • block • annotate
counters
masked
/metrics p50/p95, rule hits
Audit Log (masked snippet + hash)

Request Lifecycle

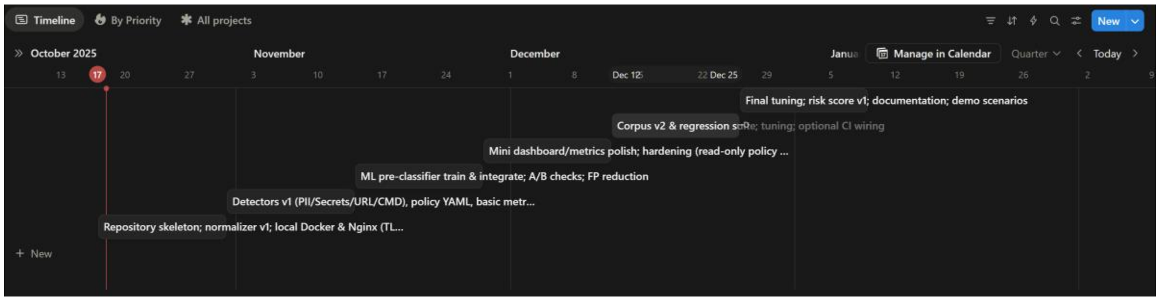


Deployment / Runtime



Client App
HTTPS 443
Nginx Container :443 -> :8080
HTTP 8080
FastAPI App Container uvicorn :8080
scrape /metrics
Prometheus/Grafana (optional external)
RO mount
volume
policy/ & configs/ read-only bind mount
logs/metrics volume
Docker Engine

Gantt Chart


Timeline
By Priority
All projects
New
October 2025
November
December
Janua
Manage in Calendar
Quarter
Today
13
17
20
27
3
10
17
24
1
8
Dec 12
22
Dec 25
29
5
12
19
26
2
9
Final tuning; risk score v1; documentation; demo scenarios
Corpus v2 & regression suite; tuning; optional CI wiring
Mini dashboard/metrics polish; hardening (read-only policy ...
ML pre-classifier train & integrate; A/B checks; FP reduction
Detectors v1 (PII/Secrets/URL/CMD), policy YAML, basic metr...
Repository skeleton; normalizer v1; local Docker & Nginx (TL...
+ New