AMIF v1.0

Autonomous Multi-Modal Intelligence Fabric

Detailed Complete Project Report

Project Type	Senior-level full-stack AI architecture and implementation project
Domain	Industrial safety, incident intelligence, predictive maintenance, operational monitoring
Core Loop	Detect -> Normalize -> Correlate -> Retrieve -> Reason -> Guard -> Alert -> Explain

AMIF v1.0 - Autonomous Multi-Modal Intelligence Fabric

Complete Project Report

Date: 2026-06-15

Project Type: Senior-level full-stack AI system design and implementation project

Primary Domain: Industrial safety, incident intelligence, predictive maintenance, and operational monitoring

1. Executive Summary

AMIF, or Autonomous Multi-Modal Intelligence Fabric, is a production-style AI incident intelligence platform. It ingests operational signals from cameras, sensors, audio streams, documents, and manual/API sources, normalizes those signals into canonical events, correlates them into incidents, retrieves relevant evidence from memory, runs an agentic investigation workflow, applies safety guardrails, and presents the result through a professional operator dashboard.

The project is designed as a resume-worthy senior full-stack and AI architecture project. The current implementation is a runnable MVP with strong architecture boundaries. Some AI services are implemented as deterministic/mock boundaries so the system can run locally without heavyweight models, but the architecture is ready to plug in real models such as YOLO, Whisper, BGE embeddings, Qdrant, LangGraph, Llama/Qwen/DeepSeek, and Llama Guard.

Core product loop:

```
Detect -> Normalize -> Correlate -> Retrieve -> Reason -> Guard -> Alert -> Explain
```

2. Problem Statement

Industrial and enterprise environments produce many isolated signals: CCTV frames, IoT sensor readings, machine logs, audio warnings, maintenance manuals, emails, and operator notes. Traditional systems usually treat these streams separately. This creates problems:

- Operators receive too many isolated alerts.
- Root-cause analysis is slow and manual.
- Evidence is scattered across cameras, sensors, manuals, and incident history.
- AI decisions are often not auditable.
- Automated actions can be unsafe without policy guardrails.

AMIF solves this by acting as an incident intelligence fabric: it fuses multimodal signals, detects patterns, retrieves supporting knowledge, generates evidence-backed investigation summaries, and routes safe actions to humans or external systems.

3. Real-World Use Cases

3.1 Factory Safety Monitoring

Detect unsafe events such as forklift activity in restricted zones, workers entering danger areas, or machinery overheating. Correlate visual detections with sensor anomalies and create high-priority incidents.

3.2 Predictive Maintenance

Monitor temperature, vibration, noise, and energy readings. When abnormal readings occur, retrieve machine manuals and past incident reports to recommend inspection steps.

3.3 Warehouse Operations

Detect unsafe vehicle movement, zone breaches, PPE violations, and abnormal activity near equipment. Automatically alert supervisors and create tickets.

3.4 Security Operations

Correlate unauthorized access, camera detections, sensor events, and operator notes. Generate evidence-backed security incident reports.

3.5 Compliance and Audit

Store all events, alerts, actions, agent runs, guard decisions, and operator acknowledgements. Provide a traceable audit trail for regulated environments.

3.6 Smart Facilities

Combine cameras, HVAC sensors, fire alarms, access systems, and maintenance documents to detect and investigate facility incidents.

4. High-Level Architecture

```
Data Sources
-> Connectors / Ingestion APIs
-> FastAPI API Gateway
-> Domain Services
- Event Service
- Vision Service
- Audio Service
- Sensor Service
- Document Service
- Search Service
- Agent Service
- Notification Service
- Action Service
- User/Auth Service
-> Redpanda-ready Event Bus Topics
-> Event Processing Layer
- validation
- deduplication
- enrichment
- anomaly generation
- time-window correlation
-> Memory Fabric
- PostgreSQL historical memory
- Redis episodic memory design
- Qdrant semantic memory design
- Neo4j-style knowledge graph endpoint
-> Agent Runtime
- Observer
- Retriever
- Investigator
- Planner
- Safety Guard
- Executor
-> Operator Console / Dashboard
```

5. Technology Stack

Backend

- Python
- FastAPI
- Pydantic
- SQLAlchemy
- JWT authentication
- RBAC middleware pattern

Storage

- PostgreSQL-compatible SQLAlchemy models
- SQLite fallback for local fast development
- Redis-ready episodic memory design
- Qdrant-ready semantic memory design
- Neo4j-style graph API

Eventing

- Redpanda included in Docker Compose
- Redpanda-ready topic contracts
- In-process event processing for MVP
- FastStream extraction path documented

AI / Intelligence Layer

- YOLO/RT-DETR-style vision service boundary
- Whisper-style audio service boundary
- OCR/document intelligence pipeline boundary
- Deterministic embedding-style semantic retrieval fallback
- Agent runtime with Observer, Retriever, Investigator, Planner, Guard, Executor
- Llama Guard-style safety policy boundary

Frontend

- Advanced vanilla JavaScript SPA
- CSS design system
- Responsive dashboard
- Animated KPI cards
- Command palette
- Interactive SVG knowledge graph
- Modals and toast notifications

Observability / Deployment

- Prometheus metrics endpoint
- Grafana and Prometheus included in Compose
- Docker Compose
- Kubernetes starter manifests

6. Implemented Project Structure

```
amif/
backend/
app/
main.py
core/
config.py
security.py
db/
session.py
init_db.py
models/
models.py
enums.py
schemas/
schemas.py
```

```
routers/
auth.py
events.py
incidents.py
alerts.py
documents.py
audio.py
actions.py
audit.py
users.py
notifications.py
knowledge.py
system.py
demo.py
health.py
services/
event_service.py
event_processor.py
document_service.py
search_service.py
agent_service.py
action_service.py
audit_service.py
static/
index.html
architecture.html
assets/
app.css
app.js
connectors/
camera_connector.py
audio_connector.py
iot_connector.py
document_connector.py
email_connector.py
workers/
event_processor/
README.md
infra/
k8s/
prometheus/
docs/
reports/
```

7. Core Backend Modules

7.1 API Gateway

The FastAPI app acts as the gateway for the current MVP. It provides authentication, routing, OpenAPI documentation, static frontend hosting, health checks, metrics, and API endpoints.

Important endpoints:

```
POST /api/auth/login
GET /api/auth/me
POST /api/events
GET /api/events
POST /api/vision/mock-detect
POST /api/audio/mock-transcribe
POST /api/sensors/temperature
GET /api/incidents
PATCH /api/incidents/{incident_id}
POST /api/incidents/{incident_id}/investigate
GET /api/incidents/{incident_id}/agent-runs
GET /api/alerts
POST /api/alerts/{alert_id}/acknowledge
POST /api/documents/upload
POST /api/search/query
GET /api/knowledge/graph
GET /api/system/stats
GET /api/audit/logs
```

7.2 Event Service

The Event Service is responsible for accepting canonical AMIF events, validating inputs, storing event metadata, logging audits, and triggering event processing.

The canonical AMIF event includes:

```
{
"schema_version": "1.0",
"event_id": "evt_xxx",
"source_id": "camera_warehouse_a_01",
"source_type": "camera",
"source_sequence": 123,
"event_type": "forklift_detected",
"severity": "medium",
"timestamp": "2026-06-01T10:30:00Z",
"tenant_id": "demo_tenant",
"location": {"site": "plant_1", "zone": "restricted_zone"},
"payload": {"object": "forklift", "confidence": 0.94},
"trace": {"producer": "vision-service"}
}
```

7.3 Event Processor

The processor handles:

- event status updates
- temperature anomaly derivation
- time-window correlation
- incident creation
- alert creation
- ticket stub creation
- agent investigation triggering

Implemented correlation rule:

```
IF forklift_detected in restricted_zone
AND temperature_anomaly occurs within 10 minutes
THEN create high-severity incident
```

7.4 Document Service

The Document Service supports text document upload, cleanup, chunking, metadata creation, and indexing into the semantic retrieval fallback.

7.5 Search Service

The Search Service implements an offline-friendly deterministic semantic retrieval system using hashing vectors and lexical overlap. It is designed to be replaced or extended with Qdrant and real embeddings.

7.6 Agent Service

The Agent Service implements a deterministic agent workflow:

```
Observer -> Retriever -> Investigator -> Planner -> SafetyGuard -> ActionExecutor
```

The agent generates:

- incident summary
- hypotheses
- evidence references
- risk score
- recommended actions
- guard decision
- trace steps

7.7 Action Service

The Action Service currently supports:

- dashboard notification stub
- ticket creation stub

- action result storage
- audit logging

It can later be extended to Jira, ServiceNow, Slack, Teams, SAP, webhooks, and email.

8. Frontend Operator Console

The AMIF frontend is a professional animated vanilla-JS SPA served by FastAPI.

Implemented Screens

1. Login
2. Overview dashboard
3. Incident command center
4. Event operations
5. Documents and RAG
6. Agent runtime
7. Knowledge graph
8. Observability
9. Security and audit

Advanced UI Features

- responsive sidebar navigation
- animated KPI counters
- command palette with Ctrl/Cmd + K
- light/dark theme toggle
- live refresh toggle
- workspace JSON export
- modal detail inspectors
- event detail modal
- graph node modal
- toast notifications
- canvas sparklines
- interactive SVG knowledge graph
- confetti animation after demo seed
- progress loading strip
- skeleton loading states
- global search

9. AI Used in the Project

Important clarification: the current MVP uses AI-ready service boundaries and deterministic mocks/fallbacks so it can run locally. It does not require heavyweight model downloads.

9.1 Vision AI

- Boundary: YOLO/RT-DETR-style object detection

- Current: mock forklift/person detection event generation
- Real integration path: YOLOv8/YOLOv11, RT-DETR, Florence-2

9.2 Audio AI

- Boundary: Whisper-style speech-to-text
- Current: mock transcription event generation
- Real integration path: Whisper, faster-whisper

9.3 Document AI

- Boundary: OCR, chunking, embedding generation
- Current: text upload, chunking, deterministic vector fallback
- Real integration path: PyMuPDF, PaddleOCR/Tesseract, BGE-M3 embeddings

9.4 RAG / Semantic Retrieval

- Current: local hashing vector search and keyword overlap
- Real integration path: Qdrant vector database, reranking, metadata filtering

9.5 Agentic AI

- Boundary: LangGraph-style agent workflow
- Current: deterministic agents for reliable local demos
- Real integration path: LangGraph plus Llama/Qwen/DeepSeek models

9.6 Safety AI

- Boundary: Llama Guard-style safety/policy validation
- Current: deterministic guard checks requiring human approval for risky actions
- Real integration path: Llama Guard or enterprise policy classifier

10. Data Flow Demo Scenario

The demo scenario works as follows:

1. User logs in as Admin.
2. User clicks Seed Demo.
3. System uploads Machine A safety manual.
4. System emits a forklift detection event from camera.
5. System emits a temperature reading from Machine A.
6. Processor derives a temperature anomaly.
7. Processor correlates forklift in restricted zone with overheating machine.
8. System creates a high-severity incident.
9. System creates an alert and ticket stub.
10. Agent runtime investigates the incident.
11. Search service retrieves relevant manual chunks.
12. Agent creates hypotheses, risk score, and action plan.
13. Safety Guard blocks actions requiring human approval.
14. Dashboard displays summary, evidence, graph, alerts, and audit trail.

11. Memory Fabric

PostgreSQL / SQLite

Used as source of truth for:

- users
- events
- incidents
- incident-event relationships
- alerts
- documents
- document chunks
- agent runs
- action results
- audit logs

Redis-ready Episodic Memory

Designed for:

- recent events
- active incidents
- agent scratchpad state
- dedupe keys
- live dashboard cache

Qdrant-ready Semantic Memory

Designed for:

- manual chunks
- safety policies
- incident summaries
- maintenance reports
- similarity search

Knowledge Graph Endpoint

The /api/knowledge/graph endpoint exposes Neo4j-style relationships:

- Incident HAS_EVIDENCE Event
- Event AFFECTS Machine
- Event LOCATED_IN Zone
- Document DESCRIBES Machine

12. Security and Governance

Implemented controls:

- JWT authentication
- password hashing
- role-based access control

- Admin, Operator, Analyst, Viewer roles
- audit logging
- action traceability
- incident update audit trail

Role capabilities:

```
Admin: full access, users, audit logs
Operator: acknowledge alerts, update incidents
Analyst: upload documents, trigger investigations
Viewer: read-only dashboard access
```

13. Observability

Implemented:

- /health endpoint
- /metrics Prometheus endpoint
- system stats endpoint
- Prometheus config
- Grafana service in Docker Compose
- dashboard observability screen

Tracked platform indicators:

- event count
- incident count
- alert count
- document count
- agent run count
- audit log count

Recommended future metrics:

- p50/p95/p99 event latency
- consumer lag
- DLQ rate
- agent latency
- RAG retrieval precision
- false alert rate
- guard rejection rate

14. Deployment

Local Run

```
cd amif
cp .env.example .env
docker compose up --build
```

Open:

```
Dashboard: http://localhost:8000
Architecture: http://localhost:8000/architecture.html
API Docs: http://localhost:8000/docs
Prometheus: http://localhost:9090
Grafana: http://localhost:3000
```

Kubernetes Readiness

Starter manifests are included for:

- namespace
- backend deployment
- backend service
- horizontal pod autoscaler

Future Kubernetes work:

- split services into separate deployments
- add ingress
- add secrets
- add network policies
- add persistent volumes
- add Helm chart

15. Current Limitations

The MVP is intentionally optimized for local portfolio demonstration.

Current limitations:

- event processing is in-process, not yet a real Redpanda consumer
- vision/audio models are mocked boundaries
- embeddings are deterministic local fallback, not real neural embeddings
- Qdrant and Redis are infrastructure-ready but not deeply integrated
- no real external Jira/Slack/ServiceNow integration yet
- no production-grade migration system yet
- no full test suite yet

16. Future Roadmap

Phase 1: Real Streaming

- Add FastStream worker
- Consume from Redpanda topics
- Add DLQ processing
- Add replay tooling

Phase 2: Real AI Models

- Add YOLO inference service
- Add Whisper/faster-whisper service
- Add BGE-M3 embeddings
- Add Qdrant indexing

Phase 3: LangGraph Agents

- Replace deterministic agent workflow with LangGraph
- Add model gateway

- Add prompt versioning
- Add model fallback

Phase 4: Enterprise Integrations

- Jira ticket creation
- Slack/Teams alerts
- ServiceNow integration
- email notifications
- SAP/CMMS hooks

Phase 5: Production Hardening

- Alembic migrations
- test suite
- OpenTelemetry distributed traces
- Kubernetes Helm chart
- load testing
- security hardening

17. Resume Positioning

Suggested resume project title:

Autonomous Multi-Modal Incident Intelligence Fabric

Suggested bullets:

- Architected and implemented an event-driven AI incident intelligence platform using FastAPI, SQLAlchemy, Redpanda-ready topic contracts, PostgreSQL, Redis/Qdrant-ready memory layers, and an advanced vanilla-JS operator console.
- Built canonical event ingestion, validation, anomaly derivation, time-window correlation, incident creation, alerting, audit logging, and agent investigation workflows.
- Designed a multi-layer memory fabric combining operational storage, semantic retrieval, episodic memory design, and a knowledge graph API.
- Implemented an auditable agent runtime with Observer, Retriever, Investigator, Planner, Safety Guard, and Executor stages.
- Developed a production-style frontend with command palette, animated KPIs, event explorer, RAG interface, agent trace viewer, interactive graph visualization, observability, and RBAC/audit screens.
- Added Docker Compose infrastructure with Postgres, Redis, Qdrant, Redpanda, Prometheus, and Grafana plus Kubernetes starter manifests.

18. Final Summary

AMIF is a complete full-stack AI architecture project that demonstrates senior-level system design, backend engineering, frontend UX, AI orchestration, event processing, memory design, governance, and deployment readiness.

The project is not just a dashboard. It is an end-to-end incident intelligence platform with clear contracts, extensible service boundaries, auditable agent workflows, and a polished operator experience.