TEAM CHARTER

Group 3, Fundamentals of Operationalizing AI, Fall 2025

EXECUTIVE SUMMARY

Our team will build a production-grade AI service for real-time volatility spike prediction in BTC/USD markets. The project transforms a research model into an operational streaming system that integrates live (or replayed) Coinbase data using Kafka, serves predictions through FastAPI, manages experiment lineage via MLflow, and monitors performance with Prometheus, Grafana, and Evidently.

This charter establishes member roles, collaboration norms, quality expectations, and ownership of critical system components. Each student is responsible for a core subsystem yet must contribute across the entire pipeline through code reviews and integration testing. The mission is to produce a reliable, observable, one-command deployable system that behaves like a real enterprise AI service.

---

1. TEAM MEMBERS & ROLE OWNERSHIP (4 MEMBERS)

Each member holds a single primary accountability area, but all team members collaborate on testing and integration. Owners ensure delivery, documentation, and troubleshooting for their subsystem.

• **Matt Ross — ML Lead**
Responsible for the final volatility model, feature schema, FastAPI inference integration, threshold decisions, and MLflow logging. Ensures the model contract remains stable and versioned through /version and /predict.

• **Asli Gulcur — Data & Ingestion Lead**
Owns Coinbase ingestion (live and replay), Kafka schema and topics, retry logic, buffering, and consumer durability. Guarantees the prediction service receives clean, time-ordered feature payloads.

• **Melissa Wong — API & Systems Lead**
Owns FastAPI logic and all production endpoints (/predict, /health, /version, /metrics). Ensures batch requests, timeout handling, graceful shutdown, load testing, and correct Prometheus metric exposure.

• **Asel Torogulova— Reliability & Monitoring Lead**
Owns CI/CD pipeline (GitHub Actions), Black/Ruff linting, integration tests, Prometheus + Grafana dashboards, Evidently drift scheduling, rollout/rollback control via MODEL_VARIANT, and operational runbook.

---

2. WORKING AGREEMENTS

Communication Standards

• Daily asynchronous check-ins via WhatsApp.
• Task planning and tracking via GitHub Projects.
• Design decisions must be recorded in a GitHub Issue (≤ 4 sentences).
• Blocking problems → escalate in chat, then commit summary to issue.

Coding & Merging Expectations

• The main branch must always stay deployable.
• All work must occur in feature branches using format:
feature/<component>-<student>
• Merge rules:
   1. Passing CI tests
   2. Peer review by one teammate
   3. Updated README if setup changes

Documentation Principles
• README must stay ≤ 10 lines (quick start only).
• All detailed docs live in docs/ and must stay concise and actionable:
team_charter.md, selection_rationale.md, slo.md, runbook.md, drift_summary.md.

---

3. DECISION-MAKING PROCESS

Decisions are owned by the accountable lead, requiring peer review for high impact changes. If disagreement persists, a 10–15 minute synchronous discussion is held. If not resolved, the most conservative operational decision wins.

Decision Area	Decision Owner	Escalation
Model, features, thresholds	ML Lead	Team vote
Kafka architecture, ingestion resilience	Data Lead	Team vote
API contract, response schema	API Lead + ML Lead	Documentation + instructor
Monitoring, rollback, dashboards	Reliability Lead	Team vote
CI failures, dependency locks	Reliability Lead	Documentation + instructor

---

4. SUBSYSTEM OWNERSHIP (DELIVERABLE MAPPING)

Deliverable	Owner
Model, feature schema, versioning, /predict consistency	ML Lead
Coinbase ingestion, Kafka pipeline, retry & buffering design	Data Lead
FastAPI services, health/version endpoints, load test	API Lead
CI/CD pipeline, Prometheus/Grafana/Evidently, rollback	Reliability Lead

---

5. PROFESSIONAL QUALITY EXPECTATIONS

• One-command startup: docker compose up -d
• Predict endpoint must return valid, versioned output under load
• System should sustain burst load (100 POST requests)
• Prometheus must expose latency, count, consumer lag, and error metrics
• Grafana must track p50/p95 latency and failure rates
• Evidently drift must run on schedule and produce a written summary
• Rollback must work using MODEL_VARIANT=baseline|ml
• Demo must show failure recovery and rollback in real time

FINAL NOTE

This charter represents professional expectations, not student conventions. Each member owns a real subsystem and is responsible for its documentation, performance, and reliability in production scenarios. The success of the project is measured not by isolated accuracy metrics, but by the operation of the end-to-end system under realistic, observable, and fault-tolerant conditions.