Production-Ready Multi-Agent Market Intelligence System — Specification (v1.0)

Goal: A compliant, high-performance, multi-agent platform for market intelligence, forecasting, and portfolio decision support with rigorous backtesting, live monitoring, and enterprise governance.

---

1. Scope & Objectives

1.1 Primary Capabilities

- Multi-agent research and analysis across equities (NASDAQ & OTC) and optional asset classes.
- Near-real-time signal generation (intraday) and EOD analytics (daily) with explainability.
- Portfolio construction and risk overlays with live & simulated execution pathways.
- Institutional-grade compliance, auditability, and model governance (SEC/FINRA/MiFID II aligned).
- Comprehensive backtesting with walk-forward, transaction-cost modeling, and stress tests.

1.2 Out of Scope (v1)

- Direct discretionary trading or automated order routing to live brokers (support via adapters/ simulation; live execution can be gated behind compliance controls in v1.1+).
- Exotic asset-class peculiarities (e.g., on-chain derivatives) beyond pluggable adapters.

1.3 Key Non-Functional Goals

- **Latency**: sub-second for streaming ingestion → features; <200ms cache lookups; <2s for insight rendering.
- **Throughput**: ≥100K msgs/min ingestion; scalable to 10M ticks/day.
- **Reliability**: SLO ≥ 99.9%; RPO ≤ 5 min; RTO ≤ 15 min.
- **Security**: Zero-trust posture, least-privilege, encrypted at rest/in transit, SBOM & signed images.
- **Portability**: OCI images, **Podman** native; Kube-compatible manifests.

---

2. Reference Architecture

2.1 High-Level Components

1. **Data Plane**
2. **Market Data Ingestors** (ticks, trades, quotes, bars); **Corporate Actions**; **Fundamentals**; **News**; **Alt-Data**.
3. **Stream Bus**: Kafka/Redpanda (exactly-once where needed), NATS for low-latency pub/sub.
4. **Feature Store**: Online (Redis/KeyDB) + Offline (PostgreSQL/TimescaleDB; parquet in S3-compatible object store).

5. **Vector Store**: pgvector/Qdrant for RAG and semantic retrieval.
6. **Intelligence Plane (Agents)**
7. Orchestrated via **A2A (Agent-to-Agent)** + **MCP (Model Context Protocol)**, tool-use & delegation.
8. Agent categories: **Ingestion**, **Quality/Lineage**, **NLP/News**, **Sentiment**, **Feature-Engineering**, **Signal-Research**, **Model-Training**, **Ensembling**, **Risk**, **Compliance**, **Backtesting/Simulation**, **Explainability**, **Governance**.
9. **Decision & Execution Plane**
10. **Portfolio Engine**: optimizer (mean-variance, risk parity, turnover & limits), scenario/VaR/ES.
11. **Execution Simulator**: limit/market/iceberg, slippage & microstructure, venue models.
12. **Broker Adapters** (paper/live with guarded release, kill-switch).
13. **Control Plane**
14. **MLOps**: experiment tracker, model registry, CI/CD, canary, drift monitors.
15. **Observability**: metrics (Prometheus), logs (Loki), traces (OpenTelemetry), alerting (Alertmanager).
16. **Compliance & Audit**: immutable logs (WORM), retention, approvals, attestations.
17. **Experience Plane**
18. **Analyst/Trader UI**: Flutter multi-platform (iOS/Android/Web/macOS/Windows/Linux).
19. **Ops/Compliance Console**: role-based dashboards; approvals; audit trails.
20. **APIs**: gRPC for low-latency; REST/GraphQL for external consumption.

2.2 Tech Choices (opinionated defaults)

- **Languages**: Rust (services/agents, execution, backtester); Dart/Flutter (front-ends); Python (research notebooks & prototypes if desired).
- **Storage**: PostgreSQL (OLTP via Supabase), TimescaleDB (time-series), ClickHouse (OLAP), Object Store (S3-compatible) for parquet, Redis for online features/caching.
- **Orchestration**: Podman pods/Compose in lower env; Kubernetes-compatible manifests for scale; HashiCorp Vault for secrets; Traefik/Envoy for gateway.
- **IaC**: Terraform + Ansible; SBOM via Syft; image signing via Cosign; SLSA provenance.

---

3. Agents & Responsibilities

Agents communicate via A2A; MCP defines tools (APIs) they may call. Each agent produces structured artifacts (schemas below) and emits events.

1. **Ingestor Agents**
2. Connectors: market data (level-1/2), fundamentals, filings (EDGAR), news (RSS/APIs), alt-data.
3. Guarantees: idempotent writes, watermarking, late-arrival handling, schema evolution.
4. **Data Quality & Lineage Agent**
5. Profiling, anomaly detection (missing fields/spikes), unit checks on distributions.
6. Lineage graph (OpenLineage), dataset SLAs, quarantine & backfill workflows.
7. **NLP / News Intelligence Agent**
8. Document unification, de-duplication, entity linking, sentiment, event extraction.
9. RAG over filings/earnings calls; topic clustering; impact scoring per ticker/sector.
10. **Feature Engineering Agent**
11. Technical factors (momentum/mean-reversion/vol), fundamental ratios, cross-sectional ranks.
12. Cross-asset & macro joins; leakage controls; feature importance diagnostics.

13. **Signal Research Agent**
14. Hypothesis generation, grid/BO tuning, episodic regime detection; emits candidate signals with metadata.
15. **Model Training Agent**
16. Pipeline for tree-based, linear, and deep (tabular + sequence + GNN if needed).
17. Walk-forward splitter; nested CV; hyper-param tuning; model cards.
18. **Ensemble & Meta-Learner Agent**
19. Stacking/blending; diversity metrics; stability regularization; turnover optimization.
20. **Risk Agent**
21. Exposure controls (sector/β/FX), VaR/ES (historical/Monte Carlo), drawdown guards, circuit breakers.
22. **Backtesting/Simulation Agent**
23. Event-driven engine; order book microstructure; borrow/short fees; corporate actions; survivorship-bias-free universes.
24. **Compliance Agent**
25. Pre-trade checks (restricted lists, holdings conflicts), post-trade surveillance, communications audit hooks, record retention tagging.
26. **Explainability Agent**
27. Per-asset and portfolio-level explanations (SHAP/Permutation), feature drift summaries, narrative generation with citations.
28. **Governance Agent**
29. Model version gates, sign-off workflows (maker/checker), deployment approvals, rollback.

---

4. Data Architecture

4.1 Schemas (select)

- **Event Envelope**

```
{
  "event_id": "uuid",
  "ts": "RFC3339",
  "source": "agent_name",
  "type": "INGEST|FEATURE|SIGNAL|RISK|PORTFOLIO|EXEC_SIM|ALERT|GOV",
  "tenant_id": "string",
  "payload": {"…": "…"},
  "lineage": {"parents": ["event_id"], "dataset_hash": "sha256"}
}
```

- **Feature Row**

```
{"asof": "ts", "symbol": "str", "feature_namespace": "str", "features":
{"f1": 0.23, "f2": -1.2}, "label": 0.004, "window": "5m", "quality": {"z":
0.8}}
```

- **Signal**

```
{"asof":"ts","symbol":"str","signal_id":"uuid","model_version":"v2025.08.1","horizon":"1d",'
0.73,"confidence":0.61,"explain":{"top_features":["mom_20","sent_pos"]}}
```

- **Portfolio Instruction (Sim/Live)**

```
{"asof":"ts","book":"core","target_weights":{"AAPL":0.02,"MSFT":
0.03},"constraints":{"gross":1.0,"net":0.5,"sector_max":
0.2},"notes":"rebalance weekly"}
```

4.2 Feature Store

- **Offline**: partitioned parquet (symbol/date), Hive-compatible metadata; tracked via data versioning (DVC/lakeFS).
- **Online**: Redis/KeyDB with TTL; write-behind to OLTP; schema registry (Protobuf/Avro) with compatibility rules.

4.3 Data Quality Gates

- Min completeness %, z-score bounds, monotonicity checks; automatic quarantine topic; repair/backfill DAG.

---

5. Modeling & Research Standards

- **Labeling**: forward returns (t+1d/1w/1m), classification (up/down) and regression (basis points).
- **Splits**: purged, embargoed K-fold CV to eliminate leakage; walk-forward with refit cadence.
- **Metrics**: Information Coefficient, Precision@K, Hit-Rate, Sharpe/Sortino, MaxDD, Turnover, Capacity.
- **Regularization**: turnover penalization, risk-budgeting, stability constraints.
- **Explainability**: global/local SHAP, ICE plots, ablation; narrative generation for UI.
- **Regime Modeling**: HMM/HSMM or clustering to adapt signal weights per regime.

---

6. Backtesting & Simulation

6.1 Engine Requirements

- Event-driven, single-source-of-truth clock; realistic order book; partial fills; queue position.
- **Costs**: commissions, spreads, slippage (square-root/impact models), borrow/short fees, taxes (configurable).
- **Constraints**: liquidity limits, ADV caps, borrow availability, hard/soft limits.
- **Corporate Actions**: splits/dividends/mergers correctly adjusted; **survivorship-bias-free** universes.
- **Scenarios**: regime shocks (e.g., 2008, 2020), circuit-breakers, halts, venue outages.

6.2 Validation Protocols

- IS/OOS separation, rolling windows; **white-paper-style** experiment manifests; seeds, hashes, reproducibility.
- Bootstrap and reality checks; deflated Sharpe; p-hacking guards.

---

7. Portfolio Construction & Risk

- **Optimizers**: mean-variance with robust covariance (Ledoit-Wolf), risk parity, Black-Litterman overlay; turnover & transaction cost terms.
- **Risk**: factor exposures (β, size, value, momentum), VaR/ES, stress; concentration, country/sector caps.
- **Policies**: kill-switches, drawdown-based de-risking, circuit breakers, hard limits enforced pre- and post-trade.

---

8. Compliance, Governance & Auditability

8.1 Core Controls

- **Recordkeeping**: WORM storage for all decisions/signals/backtests (SEC 17a-4), retention policies by jurisdiction.
- **Surveillance**: pre-trade checks (restricted lists, insider windows), post-trade analytics, exception workflows.
- **Model Risk Mgmt**: model inventory, owners, purpose, data lineage, validation reports (align with SR 11-7).
- **Explainability**: per-recommendation rationale + evidence links (filings, features, news snippets).
- **Approvals**: maker/checker with e-sign; deployment gates; change-control tickets linked to model versions.
- **Privacy**: GDPR/CCPA data subject rights; PII minimization; DLP scanners on data egress.

8.2 Access & Segregation of Duties

- RBAC/ABAC; environment isolation (dev/stage/prod); break-glass with hardware tokens; session recording for ops.

8.3 Legal & Disclaimers

- System outputs are **decision support**; no investment advice; brokerage integration requires suitability checks.

---

9. APIs & Contracts

- **gRPC Services**: low-latency feature reads, signal fetch, risk checks, sim/execution endpoints.

- **REST/GraphQL**: query portfolios, reports, audit artifacts.
- **Schema Registry**: versioned Protobuf/Avro; backward/forward compatibility tests in CI.
- **Idempotency**: request keys for writes; at-least/at-most-once semantics per endpoint.

---

10. Front-Ends (Flutter)

- **Analyst Workbench**: universe filters, feature/label explorer, model comparison, experiment timelines.
- **Signal Explorer**: per-ticker insight, confidence, rationale, linked evidence (filings/news), 'why' panels.
- **Portfolio Console**: what-if tools, optimizer controls, constraints editor, scenario runner, capacity/TC dashboards.
- **Ops/Compliance**: alerts, approval queues, audit trail viewers, retention/legal hold controls.
- **Offline-first**: caching, optimistic UI, background sync; Supabase Auth + row-level security.

---

11. Observability & SRE

- Golden signals (latency, traffic, errors, saturation) per service; SLO dashboards.
- Distributed tracing across agents; baggage for correlation IDs (model_version, experiment_id).
- Dead-letter queues; replay tools; chaos drills; disaster-recovery runbooks.

---

12. Security & Privacy

- mTLS everywhere; JWT/OIDC with short-lived tokens; confidential computing (where available).
- Secrets: Vault dynamic creds; KMS-backed encryption; envelope encryption for parquet.
- Supply chain: SBOM, image signing (Cosign), vulnerability gates; reproducible builds.
- Data minimization; masking; synthesis for dev/test.

---

13. Deployment & DevEx

- **Containers**: OCI images built with BuildKit; **Podman** pods/Compose; Kubernetes-compatible YAML for prod scale.
- **CI/CD**: GitHub Actions/GitLab CI; unit/integration/property tests; ephemeral test envs; blue/green & canary.
- **IaC**: Terraform modules (networking, DBs, object store, observability); Ansible for hosts/agents.
- **Config**: env-agnostic via declarative config; feature flags; runtime overrides with secure config maps.

---

14. Data Governance & Catalog

- Data catalog (OpenMetadata/Amundsen); ownership, SLAs, PII classification; lineage (OpenLineage).

- Change data capture (Debezium) for OLTP→warehouse; quality SLAs with alerts.

---

15. Testing Strategy

- **Unit**: Rust property tests for math/optimizers; snapshot tests for schemas.
- **Integration**: service contracts; schema compatibility; fault injection.
- **Backtest Validation**: golden datasets; expected metrics bounds; reproducibility hashes.
- **LLM/Agent**: tool-use sandboxes, prompt-injection hardening, red-team suites.
- **Performance**: soak tests; p99 latency & throughput targets; cost/perf regression guards.

---

16. Acceptance Criteria (excerpt)

- Ingestion maintains ≥99.5% completeness with <0.1% late data after 15 min.
- Feature freshness ≤ 60s for intraday factors; ≤ 5m for NLP features.
- Backtests produce reproducible manifests with hash-locked inputs; deflated Sharpe reported.
- Every signal rendered in UI includes top-k features and linked evidence.
- Compliance console shows immutable audit trail with WORM proof and retention timers.
- SLO error budget respected over rolling 28-day windows; automated rollback on breach.

---

17. Rollout Plan

- **Phase 0 (2–4 wks)**: Foundation — data bus, OLTP, object store, auth, baseline ingest, catalog.
- **Phase 1 (4–6 wks)**: Feature store, NLP agent, initial signals, backtester v1, Flutter dashboards.
- **Phase 2 (6–8 wks)**: Portfolio engine, risk/compliance agents, explainability, governance gates.
- **Phase 3 (ongoing)**: Optimization, broker adapters (paper → gated live), cost/perf hardening.

---

18. Glossary

- **A2A**: Agent-to-Agent protocol for delegation/coordination.
- **MCP**: Model Context Protocol defining tool contracts and context sharing.
- **WORM**: Write Once Read Many storage for immutable records.
- **RAG**: Retrieval-Augmented Generation (LLM + vector search/grounding).

---

19. Appendix — Example MCP Tool Specs (abbrev.)

- `fetch_market_data(symbols, start, end) -> parquet_uri`
- `quality_report(dataset_uri) -> {score, anomalies}`
- `compute_features(parquet_uri, recipe_id) -> feature_table_uri`
- `train_model(feature_table_uri, label_def, config) -> model_version`

- `score_signals(model_version, asof) -> signals_topic`
- `run_backtest(strategy_id, start, end, costs, constraints) -> report_uri`
- `pretrade_check(signals_topic, book) -> {ok|violations}`
- `publish_portfolio(targets) -> portfolio_version`

Notes

- This spec assumes **Supabase/PostgreSQL** for auth and OLTP, **Rust** for services/agents, **Flutter** for all user interfaces, and **Podman** for containerization. Substitute equivalents as needed without altering interfaces/contracts.