📡 SIGNALS: SYSTEM OPERATIONS & MAINTENANCE MANUAL

Version x.X

1. System Architecture

Signals is a distributed intelligence platform consisting of four distinct layers:

- **Persistent Storage:** Parquet-based data lake for historical pricing and supply chain entities.
- **State Management:** Redis (Message Broker) for managing background task queues.
- **Inference Engine:** Machine Learning models (XGBoost/TFT) that generate risk probabilities.
- **Dispatch:** Notification systems (Slack/HTML) that deliver actionable insights.

---

2. Infrastructure Deployment (Initial Setup)

2.1 Virtual Environment

The system must run in an isolated environment to ensure library versions remain consistent.

1. Open Terminal.
2. Navigate to the project root: `cd signals`
3. Execute setup: `python3 setup_project.py`
4. Activate environment: `source venv/bin/activate`

2.2 The Message Broker (Redis)

The autonomous features require **Redis** to be active. This acts as the system's "short-term memory."

- **Command:** `docker-compose up -d`
- **Verification:** Run `docker ps`. Look for the image `redis:alpine` running on `0.0.0.0:6379`.
- **Note:** If you are not using Docker, run `brew install redis` and then `redis-server`.

3. Autonomous Mission Profile (Always-On Mode)

Once infrastructure is verified, the system is designed to run without human intervention. To engage the autonomous cycle, you must start the **Worker** and the **Beat Scheduler**.

1. **Command:** `./start_worker.sh`
2. **Expected Log Output:**
   - o [tasks] . run_weekly_pipeline
   - o [tasks] . run_sentinel_watchdog
   - o celery@machine ready.

The Autonomous Schedule:

- **Hourly:** The **Sentinel** scans for supply chain disruptions (Factory Failures, FDA Warning Letters).
- **Weekly (Wednesday):** The system triggers the **Master Pipeline**: Ingest -> Feature Generation -> Prediction -> HTML Reporting.

4. Manual Override & Maintenance Procedures

If the autonomous scheduler is disengaged, or if an urgent out-of-cycle report is required, execute the following commands in exact order.

Step 1: Data Acquisition

Command: `python src/ingestion/nadac_ingest.py`

- **Function:** Connects to Medicaid.gov and downloads the latest National Average Drug Acquisition Cost (NADAC) file.
- **Expected Output:** A new file in `data/raw/` and updated `data/processed/nadac_history.parquet`.

Step 2: Signal Transformation

Command: `python src/features/signal_generator.py`

- **Function:** Calculates Velocity, Volatility, and Market Concentration (HHI) for every drug.
- **Verification:** Check console for "Features generated for X drugs."

Step 3: Report Synthesis

Command: `python src/reporting/generate_watchlist.py`

- **Function:** Loads the AI models, scores the new data, and builds the HTML Executive Briefing.
- **Safety Check:** If this script detects data older than 7 days, it will stamp the report with a **STALE DATA** warning.

5. Interpreting Deliverables

5.1 The Executive Risk Brief (.html)

This is the primary deliverable for management and clients.

- **The Risk Score (0.0 to 1.0):**
  - **> 0.8:** Imminent Price Spike. Immediate procurement recommended.
  - **0.5 - 0.7:** Developing Risk. Monitor daily.
- **Monopoly Index (HHI):** Measures how many companies make the drug. If this is high (e.g., > 0.6), a single factory failure will cause a total market collapse.

5.2 The Scorecard (.png)

- **Location:** reports/model_trust_score.png
- **Function:** Shows how well the AI predicted past events. If the line is trending downward, the model requires retraining.

MAINTENANCE SCHEDULE

Interval	Task	Command
Weekly	Audit Model Accuracy	python src/evaluation/scorecard.py
Monthly	Retrain AI Brain	python src/models/train_model.py
As Needed	Reset Database	docker-compose down && docker-compose up -d

APPENDIX A: COMPONENT DIRECTORY (FILE-BY-FILE EXPLANATION)

File Path	Functional Description	Execution Frequency
setup_project.py	Installs dependencies and creates the folder structure.	Once (Setup)
src/ingestion/nadac_ingest.py	Primary pricing data extractor. Connects to the Medicaid API.	Weekly
src/ingestion/sentinel_ingest.py	Scans RSS feeds and FDA sites for news-based risks.	Hourly
src/features/signal_generator.py	Turns raw prices into mathematical "signals" (trends).	Weekly
src/models/train_model.py	The logic for training the XGBoost "Price Spike" model.	Quarterly
src/tasks/celery_app.py	The air-traffic controller. Sets the automation schedule.	Always On
src/tasks/sentinel_tasks.py	Checks for supply chain events and triggers Slack alerts.	Hourly
src/reporting/generate_watchlist.py	Generates the final, human-readable HTML/CSV reports.	Weekly
src/evaluation/scorecard.py	Audits the AI by comparing old guesses to new prices.	Weekly
src/utils/notifications.py	Handles the logic for sending messages to Slack/Teams.	Event-Driven
start_worker.sh	Bash script to launch the background processes.	Always On
docker-compose.yml	Defines the Redis container configuration.	Always On

APPENDIX B: SYSTEM DIAGNOSTICS

Symptom	Probable Cause	Corrective Action
"Connection Refused" (Redis)	Redis container is not running.	Run `docker-compose up -d`.
"No Critical Risks Found"	Probability threshold is set too high.	Adjust `min_score` in `generate_watchlist.py`.
Report says "Data Stale"	`nadac_ingest` has not run recently.	Run Manual Override (Section 4).

Symptom	Probable Cause	Corrective Action
Slack alerts fail	Environment variable is missing.	Verify SLACK_WEBHOOK_URL in .env.