**GlobalPulse MVP — Technical Blueprint (v0.9)**

**Document Type:** System Specification / Whitepaper  
**Project Name:** GlobalPulse (MVP)  
**Purpose:** Build a specialized, resource-constrained platform for macroeconomic forecasting and model-agnostic LLM evaluation.  
**Audience:** Engineering teams, data scientists, economists, governance experts, product managers, and AI systems generating code from specifications.  
**License Assumption for this Document:** CC BY 4.0 for the blueprint text (implementation must respect source data licenses).  
**Change Control:** Versioned via Git; all normative requirements use **MUST**, **SHOULD**, **MAY** keywords (RFC 2119 semantics).

**1. Executive Summary**

**Problem.** Macroeconomic forecasting is diffuse across sources, units, and vintages, while LLM reasoning is difficult to evaluate beyond raw accuracy. Organizations lack a neutral, reproducible, and cost-efficient benchmark to compare how different LLMs forecast national macro indicators and to assess the coherence of their reasoning.

**Solution.** GlobalPulse provides a lean, automation-first platform that:

1. **Ingests** macro data from credible APIs (initially Trading Economics; extensible to World Bank, IMF, OECD, FRED).
2. **Normalizes** indicators into a universal taxonomy with deterministic fallbacks.
3. **Orchestrates** LLM forecasting tasks with strict vintage cutoffs and reproducible prompts.
4. **Evaluates** outputs on numerical accuracy and **coherence** (data relevance, logic, economic plausibility).
5. **Publishes** manifest-backed leaderboards, API endpoints, and dashboards with auditability, coverage, and revision awareness.

**Unique Value.**

* **Model-agnostic**: Any LLM or classical baseline can be evaluated using the same ground truth and rules.
* **Governed & Vintaged**: First-release vs latest-vintage scoring, immutable logs, and dispute resolution.
* **Startup-lean**: Modular Python/FastAPI/React stack, Redis caching, Postgres + optional object storage, containerized micro-services.
* **Automation-first**: Dynamic discovery of country indicators, GPT-assisted mapping checks, and deterministic fallbacks ensure universality without manual catalog curation.

**2. System Overview & Goals**

**2.1 What GlobalPulse Is**

* A neutral **benchmark and platform** for macroeconomic forecasting at the **country** level.
* A system to **compare** LLMs and baselines on **predictive accuracy and reasoning coherence**.
* A pipeline with **strict provenance**, **vintage control**, and **transparent manifests**.
* A **runtime and API** to run real-time tasks and historical simulations with identical rules.

**2.2 What GlobalPulse Is Not**

* Not a trading signal generator or investment advisory product.
* Not a comprehensive data warehouse attempting to mirror every statistical office.
* Not limited to a single data source; Trading Economics is the MVP ingress, not a lock-in.
* Not an LLM provider; it only orchestrates and evaluates third-party models.

**2.3 Goals**

**MVP Goals (first 6–9 months)**

* Support **5 categories** (Growth, Prices, Money, Trade, Sentiment).
* Cover **≥10 countries** initially; scale to **≥50** by month 9.
* Ingest from **Trading Economics**; optionally add **World Bank** as second source.
* Evaluate **≥3 LLMs** and **≥1 classical baseline** (e.g., ARIMA or Prophet) per category.
* Provide leaderboard, evaluation API, and manifest registry.
* Achieve end-to-end run time **≤60 minutes** for 10 countries × 5 categories × 1 horizon.

**Long-Term Goals (12–24 months)**

* **≥150 countries**, **≥3 data sources**, multi-vintage scoring (first vs latest), and rolling historical simulations.
* Advanced metrics (calibration and uncertainty where models provide intervals).
* Public reproducibility kits and peer-reviewed methodology papers.

**2.4 Stakeholder Alignment**

* **Economists**: Credible ground truth, vintage awareness, transparent fallbacks.
* **ML/AI Teams**: Consistent prompts, strict horizons, unbiased scoring, and API submissions.
* **Policy Analysts**: Country dashboards, accuracy over time, coverage and caveats.
* **Governance**: Audit trails, dispute process, hashes, and manifests.
* **Product**: Lean stack, low operational cost, rapid iteration.

**3. Core Components & Subsystems**

1. **Data Ingestion Subsystem**
   * Fetches metadata (countries, indicators) and time series.
   * Tags each datapoint with source, vintage\_date, release\_date, ingestion\_ts.
   * Handles retries, rate limiting, and caching.
2. **Taxonomy & Mapping Subsystem**
   * Deterministic mapping of country indicators to **5 canonical categories**.
   * Unit normalization and imputation rules.
   * Fallback hierarchy (primary → secondary proxies) per category.
3. **Governance Module**
   * Validates source credibility, versioning, and licensing compliance.
   * Enforces immutable records and manifest creation.
   * Runs integrity checks, quality flags, and audit logging.
4. **Forecasting Engine**
   * Orchestrates task generation, prompt packaging, and LLM calls.
   * Records predictions, reasoning, cutoff timestamps, and model metadata.
   * Supports real-time and historical simulation modes.
5. **Evaluation Framework**
   * Computes numerical metrics (MAE, RMSE, sMAPE, MAPE, Directional Accuracy).
   * Computes **Coherence Score** (Data Relevance, Logical Structure, Economic Plausibility).
   * Aggregates to model × country × category and global leaderboards.
6. **Storage Layer**
   * **Postgres** schemas for metadata, raw series, normalized series, predictions, actuals, evaluations, manifests, audits.
   * **Object storage** (MinIO/S3) for large artifacts and reasoning archives.
7. **API & Reporting Interface**
   * FastAPI endpoints for forecasts, actuals, evaluations, manifests, coverage.
   * React dashboard (leaderboards, coverage heatmaps, time series, forecast cards).
8. **Orchestration & Scheduling**
   * Prefect or Airflow for ETL and evaluation schedules.
   * Redis for queues and caching; RQ or Celery for lightweight workers.

**4. Data Flow & Architecture**

**4.1 End-to-End Flow (Text Diagram)**

[External APIs]

| \

| \ TradingEconomics / WorldBank / IMF / FRED

v

+--------------------------------------+

| Ingestion Adapters (Python) |

| - metadata (countries, indicators) |

| - time series (history, latest) |

| - rate limit & retries |

+-------------------+------------------+

v

+--------------------+

| Validation/Mapping |

| - unit normalize |

| - taxonomy map |

| - fallbacks |

| - quality flags |

+----+-----------+---+

| |

v v

+---------------+ +----------------+

| Raw Storage | | Normalized TS |

| (Postgres) | | (Postgres) |

+-------+-------+ +--------+-------+

| |

| vintages |

v v

+--------------------------------+

| Governance & Manifest Builder |

| - freeze configs & hashes |

| - audit logs |

+----------------+---------------+

|

v

+--------------------+

| Forecast Orchestr. |

| - task builder |

| - model connectors |

+----+-----------+---+

| |

v v

+-----------+ +--------------+

| Predictions| | Reasoning |

| (Postgres) | | (Object st.)|

+-----+------+ +------+------+

| |

v |

+---------------------------------+

| Evaluation Framework |

| - numerical metrics |

| - coherence scoring |

+------------------+--------------+

|

v

+------------------+

| Evaluations DB |

+---------+--------+

|

v

+---------------------+

| API & Dashboards |

| - leaderboards |

| - coverage, charts |

+---------------------+

**4.2 Execution Modes**

* **Real-time**: Forecast future period; wait for ground truth to arrive; then score.
* **Historical simulation**: “Pretend date” cutoff; allow full backtest quickly.

**5. Functional Requirements (Normative)**

**FR-1 Data Discovery**

* The system **MUST** fetch the list of supported countries and indicators from at least one API at startup and refresh it daily.
* The system **MUST** validate user-selected {country, indicator} pairs against the metadata list before any query.
* The system **MUST** cache metadata in Redis with a TTL of 24 hours and fall back to the last valid cache on upstream failure.

**FR-2 Data Ingestion**

* The system **MUST** ingest historical time series for requested {country, indicator} with a configurable window (default 5 years).
* Each ingested record **MUST** include date, value, unit, source, source\_url, release\_date (if available), vintage\_date, ingestion\_ts.
* The system **MUST** respect provider rate limits using token buckets and exponential backoff.

**FR-3 Normalization & Taxonomy**

* The system **MUST** convert units to canonical units per indicator definition (e.g., % y/y, USD millions, index(=100)).
* The system **MUST** deterministically map indicators into one of five categories and record the mapping decision.
* If a primary indicator is missing, the system **MUST** apply a documented fallback in ranked order and record the substitution with reason.

**FR-4 Governance & Provenance**

* The system **MUST** store all records immutably; updates are appended as new vintages.
* The system **MUST** compute and store a SHA-256 content hash per dataset snapshot and manifest.
* The system **MUST** maintain an audit log of ingestion and evaluation runs.

**FR-5 Forecast Orchestration**

* The system **MUST** construct a forecast task with {country, category, target\_indicator, horizon, cutoff\_date, context\_data} and a standardized prompt.
* The system **MUST** submit tasks to at least 3 LLM endpoints in MVP (e.g., GPT, Gemini, Grok), configurable via secrets.
* The system **MUST** store the full model response, including numeric prediction(s) and reasoning text, with timestamps and model version.

**FR-6 Evaluation**

* The system **MUST** compute MAE, RMSE, sMAPE, MAPE, and Directional Accuracy for each prediction when ground truth is available.
* The system **MUST** compute a Coherence Score per prediction using the specified rubric.
* The system **MUST** support first-release vs latest-vintage scoring modes.

**FR-7 API & Reporting**

* The system **MUST** expose read-only endpoints for forecasts, actuals (with vintages), evaluations, coverage, and manifests.
* The system **MUST** provide filtering by {model, country, category, horizon, date\_range, vintage\_mode}.
* The system **MUST** render a leaderboard and coverage heatmap in a React dashboard.

**FR-8 Performance**

* The system **MUST** serve metadata and coverage queries in **≤1 second** p95 under 100 concurrent users with dataset sizes specified in NFR-Scalability.
* The system **MUST** complete the MVP monthly evaluation cycle (10 countries × 5 categories × 1 horizon × 4 models) within **≤60 minutes** on a single modest VM (e.g., 4 vCPU, 16 GB RAM) excluding external LLM latency.

**FR-9 Security & Access**

* The system **MUST** store all API keys encrypted at rest and never expose them via logs.
* The system **MUST** implement RBAC with roles {admin, curator, evaluator, viewer}.
* The public API **MUST** be read-only; mutations restricted to internal services.

**FR-10 Reproducibility**

* Every evaluation run **MUST** produce a manifest including model versions, data vintages, parameters, and hashes.
* The same manifest **MUST** reproduce identical scores when re-executed.

**6. Non-Functional Requirements (NFRs)**

**NFR-Performance**

* p95 latency for metadata endpoints: ≤1 second.
* p95 latency for evaluations query (aggregate): ≤2 seconds with Redis caching.
* Background ETL throughput: ≥5,000 records/minute sustained on MVP VM with network-bound constraints.

**NFR-Scalability**

* Support **≥50 countries** and **≥500 indicators** within 12 months.
* Handle **≥10 million rows/year** of time series with Postgres partitioning by {country, indicator} and monthly ranges.
* Horizontal scale using container replicas for API and workers.

**NFR-Security**

* Secrets in environment variables or secret manager; AES-256 at rest; TLS in transit; IP allow-listing for internal write endpoints.

**NFR-Integrity & Availability**

* Daily offsite snapshots of Postgres and object storage; RPO ≤24 hours.
* Scheduled integrity checks of hashes weekly; alert on mismatch.

**NFR-Maintainability**

* 80% unit test coverage for transformation and scoring logic.
* CI pipeline with linting, type checks (mypy), and integration tests on seed datasets.
* Twelve-factor app principles for services.

**NFR-Compliance**

* Respect data source licenses; store attribution per series; block re-distribution where prohibited in public API; allow metadata exposure and derived metrics when permitted.

**7. Evaluation Methodology**

**7.1 Targets and Horizons**

* **Targets** per category (canonical):
  + **Growth:** Real GDP growth (% y/y), quarterly.
  + **Prices:** CPI inflation (% y/y), monthly.
  + **Money:** Policy rate (%), monthly average or end-of-month.
  + **Trade:** Trade balance (USD millions), monthly.
  + **Sentiment:** Consumer confidence index (base = country-specific standard), monthly.
* **Horizons:** Next observable period for each frequency (e.g., next month for CPI, next quarter for GDP). Historical simulation uses any past period with known actuals.

**7.2 Numerical Metrics**

Let y^t\hat{y}\_ty^​t​ be the forecast for period ttt, yty\_tyt​ the realized value.

* **MAE** =1n∑t=1n∣y^t−yt∣= \frac{1}{n}\sum\_{t=1}^{n} |\hat{y}\_t - y\_t| =n1​∑t=1n​∣y^​t​−yt​∣
* **RMSE** =1n∑t=1n(y^t−yt)2= \sqrt{\frac{1}{n}\sum\_{t=1}^{n} (\hat{y}\_t - y\_t)^2} =n1​∑t=1n​(y^​t​−yt​)2​
* **MAPE** =100n∑t=1n∣y^t−ytyt∣= \frac{100}{n}\sum\_{t=1}^{n} \left|\frac{\hat{y}\_t - y\_t}{y\_t}\right| =n100​∑t=1n​​yt​y^​t​−yt​​​ (excluded when yt=0y\_t=0yt​=0; if zeros occur, use sMAPE)
* **sMAPE** =100n∑t=1n∣y^t−yt∣(∣y^t∣+∣yt∣)/2= \frac{100}{n}\sum\_{t=1}^{n} \frac{|\hat{y}\_t - y\_t|}{(|\hat{y}\_t|+|y\_t|)/2} =n100​∑t=1n​(∣y^​t​∣+∣yt​∣)/2∣y^​t​−yt​∣​
* **Directional Accuracy** = fraction where sign(yt−yt−1)=sign(y^t−yt−1)\text{sign}(y\_t - y\_{t-1}) = \text{sign}(\hat{y}\_t - y\_{t-1})sign(yt​−yt−1​)=sign(y^​t​−yt−1​).

**Normalization for Leaderboards**

* Compute **z-scores** within each {country, category} across models to mitigate scale effects, then map to **0–100** using min–max on z-scores clipped to [−2, +2].
* Composite numerical score **NumAcc** is the average of normalized RMSE and sMAPE complements (100 − score).

**7.3 Coherence Scoring (0–100)**

**Components and Weights:**

* **Data Relevance (40%)**: References to provided main target and at least one correlate; numeric mentions within ±10% of provided values score full credit.
* **Logical Structure (40%)**: Stepwise reasoning without contradiction; explicit “data → analysis → conclusion” or numbered steps.
* **Economic Plausibility (20%)**: Alignment to rule set per category (e.g., rising oil prices likely increase inflation in oil-importing countries).

**Implementation:**

* Rule-based extraction of indicator mentions and numeric values.
* Optional LLM-as-judge ensemble with blind prompts; majority vote; threshold calibration against human-scored samples (≥5% sample rate).

**Final Composite Score:**

Score=0.7×NumAcc+0.3×Coherence\text{Score} = 0.7 \times \text{NumAcc} + 0.3 \times \text{Coherence}Score=0.7×NumAcc+0.3×Coherence

**7.4 Example Test Case (Inflation, Brazil)**

* **Input:** Monthly CPI y/y 2019-01 to 2025-06, correlates: Core CPI, PPI, Brent crude USD, BRL/USD.
* **Task:** Predict 2025-07 CPI y/y.
* **Model Output:** 4.5% with reasoning referencing recent core CPI decline and stable currency.
* **Ground Truth:** 4.7%.
* **Metrics:** MAE=0.2, RMSE=0.2, sMAPE≈4.26. Directional accuracy = 1 if CPI rose vs previous month and forecast implied rise.
* **Coherence:** Data Relevance=90, Logical Structure=90, Economic Plausibility=100 → Coherence=92.
* **Composite:** NumAcc normalized to, say, 88 → Final=0.7×88+0.3×92=89.2.

**8. Indicator Taxonomy & Country-Specific Mapping**

**8.1 Canonical Categories, Primaries, Fallbacks**

**Growth**

* **Primary:** Real GDP growth (% y/y, quarterly).
* **Fallbacks (ranked):** Industrial production y/y (%), PMI output index level, Employment growth y/y (%).

**Prices**

* **Primary:** CPI inflation y/y (%).
* **Fallbacks:** Core CPI y/y (%), PPI y/y (%), GDP deflator y/y (%).

**Money**

* **Primary:** Central bank policy rate (%).
* **Fallbacks:** 3-month interbank rate (%), 10-year government bond yield (%), M2 growth y/y (%).

**Trade**

* **Primary:** Trade balance (USD millions).
* **Fallbacks:** Exports (USD millions) and Imports (USD millions) to compute balance; Current account balance (% of GDP).

**Sentiment**

* **Primary:** Consumer confidence index (country baseline).
* **Fallbacks:** Business confidence index, Composite PMI.

**8.2 Indicator Universe (Explicit Examples)**

* **Growth:** Real GDP growth; GDP constant prices; Industrial Production index; PMI Output; Employment level and rate; Capacity utilization.
* **Prices:** CPI headline; Core CPI; PPI; GDP deflator; Food CPI; Energy CPI; Import prices.
* **Money:** Policy rate (e.g., repo, Fed Funds, Selic); Interbank 3M; 10Y yield; M2; M3; Credit to private sector.
* **Trade:** Exports; Imports; Trade balance; Current account; Terms of trade index; FX reserves.
* **Sentiment:** Consumer Confidence; Business Confidence; PMI Composite; PMI Manufacturing; PMI Services.

**8.3 Dynamic Discovery & Mapping Algorithm (Deterministic)**

1. Query /countries and /indicators/country/{country}.
2. Normalize indicator names via rule set (regex and alias mapping).
3. For each category, attempt primary; if unavailable or coverage <80% over lookback window, escalate to fallback 1, then fallback 2, etc.
4. Persist mapping decision with decision\_reason and coverage\_ratio.
5. Validate unit; if non-canonical, convert using configured transformation.
6. Emit mapping JSON:

{

"country": "Brazil",

"category": "Prices",

"primary": "CPI Inflation Rate",

"fallback\_used": false,

"coverage\_ratio": 0.98,

"unit": "% y/y",

"correlates": ["Core CPI", "PPI", "Brent USD"]

}

**8.4 Missing or Conflicting Data Handling**

* **Missing values within series:** Linear interpolation only for gaps ≤2 consecutive periods; otherwise mark missing and lower quality flag.
* **Conflicts between sources:** Prefer official source; if unavailable, use aggregator with lower trust score; record both with provenance.
* **Sparse series rule:** Exclude series with <10 observations in last 5 years for forecasting tasks.

**9. Data Governance & Integrity**

**Source Credibility Policy**

* Tier A: Official national statistics offices and central banks.
* Tier B: Multilaterals (IMF, World Bank, OECD, BIS).
* Tier C: Aggregators with documented methodologies (e.g., Trading Economics).
* For public leaderboards, **MUST** use Tier A or B when available; Tier C allowed only when A or B missing.

**Versioning & Vintages**

* All time series **MUST** store vintage\_date. First release identified as the earliest vintage for a period.
* Evaluation **MUST** support:
  + **First-release mode** (timeliness realism).
  + **Latest-vintage mode** (stability check).

**Auditability**

* Every ingestion batch and evaluation run produces:
  + **Manifest** with hashes of input datasets, model versions, parameters, and code commit SHA.
  + **Audit log** with timestamp, operator (service identity), and outcome.

**Conflict Resolution**

* Disputes about data or scoring: 3-tier process (technical review → governance council vote → published resolution).
* All resolutions **MUST** be linked to the affected manifest IDs.

**Licensing & Attribution**

* Store license, terms\_url, and attribution per series.
* Public API **MUST NOT** redistribute raw values where license prohibits; in such cases provide coverage, metadata, and derived metrics only.

**10. System Architecture**

**10.1 Services**

* **ingestor** (Python): Pulls metadata and series; writes raw tables; respects rate limits.
* **normalizer** (Python): Unit conversions, taxonomy mapping, fallbacks; writes normalized tables.
* **governor** (Python): Integrity checks, manifest builder, policy enforcement.
* **forecaster** (Python): Task builder, prompt packager, LLM connectors; writes predictions.
* **evaluator** (Python): Numerical metrics, coherence scoring; writes evaluations.
* **api** (FastAPI): Read-only endpoints.
* **dashboard** (React + Tailwind): Leaderboards and charts.
* **worker** (RQ/Celery): Queued jobs for forecasting and evaluation.
* **scheduler** (Prefect/Airflow): Orchestrates periodic runs.

**10.2 Technology Choices**

* **Python 3.11**, **FastAPI**, **Pydantic**, **SQLAlchemy**.
* **Postgres 15+** with native partitioning.
* **Redis** for caching and queues.
* **MinIO or S3** for object storage.
* **Docker** for containers; **docker-compose** or **Kubernetes** for orchestration (K8s optional for MVP).
* **GitHub Actions** for CI/CD.

**10.3 Database Schemas (Key Tables)**

**sources**

* source\_id (pk), name, domain, tier, license, terms\_url.

**countries**

* country\_code, country\_name, region, income\_group.

**indicators**

* indicator\_id (pk), name, unit\_canonical, category, primary\_rank (0=primary, 1=fallback1, ...), aliases (jsonb).

**series\_raw** (partitioned by country\_code, indicator\_id, year)

* series\_id (pk), country\_code, indicator\_id, date, value\_raw, unit\_raw, source\_id, source\_url, release\_date, vintage\_date, ingestion\_ts, hash.

**series\_norm** (partitioned)

* norm\_id (pk), series\_id, value, unit\_canonical, quality\_flag, transform\_notes.

**taxonomy\_mapping**

* map\_id (pk), country\_code, category, primary\_indicator\_id, fallback\_indicator\_id, reason, coverage\_ratio, decision\_ts.

**manifests**

* manifest\_id (pk), created\_ts, code\_commit\_sha, input\_hashes (jsonb), params (jsonb), vintage\_mode.

**models**

* model\_id (pk), name, vendor, version, context\_limit, notes.

**forecast\_tasks**

* task\_id (pk), country\_code, category, target\_indicator\_id, target\_period, cutoff\_ts, manifest\_id, context\_ref.

**predictions**

* prediction\_id (pk), task\_id, model\_id, prediction\_value, prediction\_unit, issued\_ts, reasoning\_uri, raw\_response\_uri.

**actuals**

* actual\_id (pk), country\_code, indicator\_id, target\_period, value, unit, release\_date, vintage\_date, source\_id.

**evaluations**

* eval\_id (pk), prediction\_id, actual\_id, mae, rmse, mape, smape, directional\_acc, coherence, score\_composite, run\_id, vintage\_mode.

**audits**

* audit\_id, entity\_type, entity\_id, action, ts, actor, details.

**Indexes**

* Composite on {country\_code, indicator\_id, target\_period} for actuals and predictions.
* B-tree on {model\_id, country\_code, category} for evaluations.
* GIN indexes for JSONB fields (input\_hashes, params, aliases).

**11. API Specification (Read-Only, MVP)**

**General**: JSON responses; pagination via limit and cursor; vintage\_mode ∈ {first\_release, latest}.

1. GET /v1/coverage?country=BRA
   * **200**: { "country": "BRA", "categories": [{"name":"Prices","primary":"CPI","coverage":0.98,"fallback\_used":false}, ...] }
2. GET /v1/series?country=BRA&indicator=CPI&start=2020-01-01&end=2025-12-31&vintage\_mode=latest
   * **200**: { "meta": {...}, "data": [{"date":"2024-12-01","value":4.7,"unit":"% y/y","vintage\_date":"2025-01-10"}, ...] }
3. GET /v1/forecasts?country=BRA&category=Prices&horizon=2025-07
   * **200**: list of forecasts with model\_id, prediction\_value, issued\_ts, manifest\_id.
4. GET /v1/evaluations?model=GPT5&country=BRA&category=Prices&date\_from=2025-01&date\_to=2025-12&vintage\_mode=first\_release
   * **200**: metrics and composite scores per prediction, with links to manifests.
5. GET /v1/manifests/{manifest\_id}
   * **200**: full manifest JSON with hashes and parameters.
6. GET /v1/leaderboard?group\_by=category&vintage\_mode=first\_release&period=2025Q1-2025Q4
   * **200**: ranked models by composite score with coverage weights and confidence intervals where applicable.

**Error Codes**

* **400** invalid params; **404** not found; **429** rate limited; **500** internal error.
* Error payload includes error\_code, message, hint.

**12. Forecasting Engine: Prompts & Tasking**

**12.1 Standardized Prompt Template (Deterministic)**

You are forecasting the {target\_indicator} for {country} for period {target\_period}.

Use only the provided data. Do not introduce external facts.

MAIN SERIES (canonical unit: {unit}):

{json\_main\_series\_last\_60\_points}

CORRELATES:

{json\_correlates\_last\_60\_points}

TASK:

1) Provide a single numeric prediction for {target\_period} in {unit}.

2) Provide step-by-step reasoning referencing specific data points and dates.

3) List explicit assumptions or caveats.

4) Do not include ranges unless asked. Output JSON in the following schema:

{

"prediction\_value": <number>,

"unit": "<unit>",

"reasoning": "<200-500 words, stepwise>",

"assumptions": ["...","..."]

}

**Constraints**

* Truncate context to last 60 points per series or fewer if model context limit is small.
* Reasoning length bounded to ensure parsability.

**12.2 Task Lifecycle**

1. **Build** task from normalized data at cutoff.
2. **Select** models based on availability and cost budget.
3. **Submit** to LLM connectors with retry on 5xx and rate throttling.
4. **Validate** JSON schema of output; reject and resubmit once if malformed.
5. **Persist** prediction and reasoning artifacts (object storage).
6. **Schedule** evaluation when actuals arrive or in simulated mode immediately.

**13. Innovation & Resource Strategy**

* **Open-first**: Prefer World Bank and IMF where licensing enables redistribution; use Trading Economics for breadth in MVP with caching.
* **Cost Guardrails**:
  + Daily ETL within 100–500 API calls using batch endpoints and country-level fetches.
  + LLM calls capped per cycle (configurable), e.g., 10 countries × 5 categories × 3 models = 150 calls per run.
* **Automation-first**:
  + GPT-assisted alias discovery to map local indicator names to canonical forms, with human-review queue for conflicts.
  + Auto-generated coverage dashboards and anomaly alerts.
* **Lean Infra**:
  + Single VM deployment (e.g., 4 vCPU/16 GB RAM + managed Postgres) for MVP; optional scale-out later.
  + Dockerized services for portability; IaC via Terraform when multi-env is needed.

**14. Cross-Role Dependencies (Explicit)**

1. **Data Logistics & Taxonomy → Evaluation Methodology**
   * Supplies canonical targets, correlates, and fallbacks per country.
   * Evaluation relies on consistent units and mappings.
2. **Evaluation Methodology → Governance**
   * Defines scoring formulas; Governance versions and freezes them.
   * Dispute resolution uses manifests referencing these rules.
3. **Governance → System Architecture**
   * Imposes vintage storage, immutability, audit logs, and RBAC; Architecture implements constraints.
4. **System Architecture → Predictive Data Analyst**
   * Provides data access patterns and context windows; Analyst designs realistic horizons and correlates.
5. **Innovation/Resource Strategy → Architecture & Ops**
   * Cost caps for LLM calls and API usage; Architecture enforces quotas and caching.
6. **Critical Collaborator → All**
   * Conducts assumption checks, identifies risks, proposes alternatives, and triggers design updates via change requests.

**15. Risks, Trade-offs, and Alternatives**

**Risk 1: Indicator inconsistency across sources**

* **Mitigation:** Deterministic aliasing, unit tests on conversions, tiered source policy, and provenance retention.
* **Trade-off:** Extra latency and complexity vs faster naive merges.

**Risk 2: LLM hallucinations and fabricated numbers**

* **Mitigation:** Prompt constraint “use only provided data,” schema validation, coherence scoring penalties, and random human audits.
* **Alternative:** Disallow free-text reasoning and require structured “evidence lines.”

**Risk 3: API rate limits and outages**

* **Mitigation:** Redis caching, nightly prefetch, and degraded mode using last snapshot.
* **Alternative:** Mirror to object storage periodically within license limits.

**Risk 4: Vintage revisions altering scores post-hoc**

* **Mitigation:** Dual scoring modes; manifests pin first vs latest; public changelog.

**Risk 5: Coverage bias toward data-rich countries**

* **Mitigation:** Coverage weighting in global scores; regional leaderboards; minimum coverage thresholds.

**Risk 6: Cost overruns for LLM inference**

* **Mitigation:** Cap calls; batched horizons; selective model evaluation; periodic instead of continuous runs.

**Risk 7: Governance capture by single funder or vendor**

* **Mitigation:** Diverse steering council; disclosed conflicts; published decisions.

**16. Constraints**

**Technical**

* Languages: Python 3.11 for backend; TypeScript/React for frontend.
* Frameworks: FastAPI, Pydantic, SQLAlchemy, Prefect/Airflow, Redis, Postgres.
* Avoid vendor-locked proprietary orchestration for MVP.
* Logging: structured JSON with OpenTelemetry-compatible fields.

**Organizational**

* Small team assumption: 2 backend, 1 data engineer, 1 economist, 1 frontend, 0.5 devops.
* Documentation: Markdown + MkDocs for internal docs; OpenAPI spec auto-generated.

**Cost**

* Monthly cloud budget target: **≤ USD 2,000** for MVP environment.
* LLM spend cap per monthly run: **≤ USD 500**.

**Governance & Compliance**

* Only Tier A/B/C sources with documented licensing.
* Public redistribution restricted by license; metadata always allowed.

**17. Success Criteria (Testable)**

**SC-1 Ingestion & Normalization**

* Ingest and normalize **≥100 indicators** across **≥10 countries** with **≥95% taxonomy accuracy** (verified via spot audit of 100 mappings).
* Median record normalization time **≤200 ms** per row in batch.

**SC-2 Forecast & Evaluate**

* Produce forecasts for **5 categories** across **≥10 countries** with **≥3 LLMs** and **≥1 baseline** in a single run.
* Complete end-to-end run within **≤60 minutes** excluding external LLM latency; **≤120 minutes** including LLM calls capped at 150.

**SC-3 Coherence & Leaderboard**

* Compute coherence for **100%** of predictions; inter-rater correlation **≥0.7** between automated and human sample on 50 predictions.
* Publish leaderboard with composite scores and coverage weighting; API returns results within **≤2 seconds** p95.

**SC-4 Reproducibility**

* Re-running an evaluation with the same manifest yields identical scores (**bit-for-bit** equality on JSON outputs).

**SC-5 Governance**

* All ingestions and evaluations logged with hashes; weekly integrity job passes **100%** hash checks for prior week artifacts.

**18. Deliverables (MVP)**

1. **Code Repositories**
   * globalpulse-ingest (adapters, normalization, mapping rules).
   * globalpulse-core (schemas, governance, forecasting, evaluation).
   * globalpulse-api (FastAPI service with OpenAPI spec).
   * globalpulse-ui (React dashboard).
   * globalpulse-infra (Dockerfiles, compose, IaC if used).
2. **Data Artifacts**
   * Postgres database with schemas and seed fixtures.
   * Object storage bucket with reasoning texts and raw model responses.
   * Example manifests for real-time and historical runs.
3. **Documentation**
   * Architecture overview diagrams and ADRs (Architecture Decision Records).
   * API reference (OpenAPI JSON + HTML).
   * Governance Charter and Dispute Process.
   * Operations runbooks for ETL, evaluation, and release.
4. **Test Suites**
   * Unit tests for conversions and mapping.
   * Integration tests for end-to-end run using synthetic data.
   * Coherence calibration set with human labels.
5. **Dashboards**
   * Leaderboard view (global, per country, per category).
   * Coverage heatmap.
   * Forecast vs actual time-series view.
   * Integrity status (hash checks, ingestion health).

**19. Implementation Details**

**19.1 ETL Pseudocode (Normalization & Mapping)**

def normalize\_and\_map(country\_code: str, lookback\_years: int = 5):

meta = fetch\_metadata(country\_code) # cached

mappings = {}

for category in CATEGORIES:

primary = select\_primary\_indicator(meta, category)

series = fetch\_series(country\_code, primary) if primary else None

if not is\_usable(series):

for fb in FALLBACKS[category]:

fb\_series = fetch\_series(country\_code, fb)

if is\_usable(fb\_series):

mappings[category] = (fb, "fallback", coverage(fb\_series))

series = fb\_series

break

else:

mappings[category] = (primary, "primary", coverage(series))

if not series:

record\_unavailable(country\_code, category)

continue

norm = convert\_units(series, CANONICAL\_UNITS[category])

quality = quality\_flags(norm)

store\_raw\_and\_norm(series, norm, quality)

persist\_mapping\_decisions(country\_code, mappings)

**19.2 Coherence Scoring Pseudocode**

def coherence\_score(reasoning\_text: str, payload: dict, category: str) -> float:

dr = score\_data\_relevance(reasoning\_text, payload) # 0..100

ls = score\_logical\_structure(reasoning\_text) # 0..100

ep = score\_economic\_plausibility(reasoning\_text, category) # 0..100

return 0.4\*dr + 0.4\*ls + 0.2\*ep

**19.3 Evaluation Job Outline**

def evaluate\_prediction(prediction\_id: str, vintage\_mode: str):

pred = db.get\_prediction(prediction\_id)

actual = db.get\_actual(pred.country, pred.indicator, pred.target\_period, vintage\_mode)

metrics = {

"mae": abs(pred.value - actual.value),

"rmse": sqrt((pred.value - actual.value)\*\*2),

"mape": mape(pred.value, actual.value),

"smape": smape(pred.value, actual.value),

"directional\_acc": directional(pred.value, actual.prev\_value, actual.value)

}

coh = coherence\_score(load\_reasoning(pred.reasoning\_uri), build\_payload(pred), pred.category)

composite = 0.7 \* normalize\_numerical(metrics) + 0.3 \* coh

db.insert\_evaluation(prediction\_id, actual.id, metrics, coh, composite, vintage\_mode)

**20. Testing & QA Plan**

**Unit Tests**

* Unit conversion functions for each indicator type.
* Mapping rules for alias detection and category assignment.
* Coherence component scorers with canned texts.

**Integration Tests**

* Synthetic dataset with controlled values and drift.
* End-to-end run from ingestion to leaderboard with fixed random seeds and mocked LLM.

**Backtesting Validation**

* Compare baseline ARIMA to LLMs on 12 months of CPI for 10 countries; assert directional accuracy differences and sanity thresholds.

**Data Integrity**

* Hash verification of raw vs normalized pipelines.
* Vintage consistency checks: first release ≤ latest in presence of revisions as per source semantics.

**Performance**

* Load tests on API endpoints with 100 concurrent clients.
* Batch ETL throughput benchmarks.

**21. Change Management**

* **Methodology versions** are semantically versioned: methodology@1.0.0.
* **Breaking changes** require deprecation notices and coexistence windows.
* **Manifests** reference exact methodology versions and code commits.

**22. Operational Runbook (MVP)**

* **Nightly 02:00 UTC**: Refresh metadata, prefetch series for covered countries.
* **Monthly T-2d**: Freeze forecast run manifest; generate tasks.
* **Monthly T-1d**: Execute forecasting tasks; store predictions and reasoning.
* **When actuals arrive**: Trigger evaluation jobs per indicator and country.
* **Weekly**: Publish dashboard refresh; run integrity hash checks; rotate logs and snapshots.

**23. Security Model**

* **RBAC Roles**
  + admin: full control.
  + curator: approve mappings and resolve conflicts.
  + evaluator: run evaluations and publish manifests.
  + viewer: read-only API and dashboard.
* **Secrets**: Stored in a secret manager or .env encrypted at rest; rotated quarterly.
* **Network**: API behind WAF and rate limiter; internal services on private network.
* **Logging**: No PII; redact tokens; emit sign-in, write, and manifest events to audit log.

**24. Example Manifests**

{

"manifest\_id": "m-2025-08-27-rt-01",

"created\_ts": "2025-08-27T08:00:00Z",

"vintage\_mode": "first\_release",

"countries": ["USA","BRA","IND","NGA","ZAF","MEX","GBR","DEU","JPN","CHN"],

"categories": ["Growth","Prices","Money","Trade","Sentiment"],

"horizon\_policy": "next\_period",

"lookback\_years": 5,

"models": [

{"name":"GPT-5","vendor":"OpenAI","version":"5.0.1"},

{"name":"Gemini-2","vendor":"Google","version":"2.1"},

{"name":"Grok-3","vendor":"xAI","version":"3.0"},

{"name":"ARIMA","vendor":"baseline","version":"0.2"}

],

"code\_commit\_sha": "d34db33f",

"input\_hashes": {

"series\_norm": "sha256:...",

"taxonomy\_mapping": "sha256:...",

"prompts": "sha256:..."

},

"params": {

"context\_points": 60,

"coherence\_weights": {"data\_relevance":0.4,"logical\_structure":0.4,"economic\_plausibility":0.2},

"composite\_weights": {"numerical":0.7,"coherence":0.3}

}

}

**25. Roadmap (MVP → v1)**

* **Month 1–2**: Core schemas, Trading Economics adapter, normalization and taxonomy, seed dashboard.
* **Month 3–4**: Forecasting engine; 3 LLM connectors; baseline model; first simulations.
* **Month 5–6**: Coherence scoring calibration; public read-only API; governance charter.
* **Month 7–9**: Add second data source (World Bank); expand to 50 countries; publish first annual report.

**26. Conclusion**

GlobalPulse delivers a rigorous, vintaged, and model-agnostic benchmark for macroeconomic forecasting with a lean, automation-first design. By combining numerical accuracy with a principled coherence score and by enforcing governance through manifests and audits, it creates trustworthy visibility into how LLMs reason about—and predict—core economic dynamics. The MVP is intentionally minimal yet complete: it can run on a single VM, evaluate multiple models fairly, publish a reproducible leaderboard, and scale to broader coverage and richer metrics without architectural rewrites.

**Next Steps:**

* Implement ingestion and normalization with Trading Economics metadata.
* Stand up Postgres schemas and Redis cache.
* Build the forecasting orchestrator and one LLM connector end-to-end.
* Implement evaluation and coherence pipeline with calibration set.
* Launch the read-only API and minimal dashboard; iterate toward v1 per roadmap and success criteria.