If you'd like, next we can:

- Design **Phase 5: Explanation / Q&A Agent**, where someone can ask:
  - "Why did this trade fail?"
  - "What are the top reasons for validation failure today?"
  - "Summarize the reporting status for venue XNAS."

That's where we plug in the **LLM conversational layer** on top of all the artifacts you just built.

give complete steps that what we did now

Below is a **clean, complete, end-to-end summary** of *everything you have built so far* — exactly what you need for documentation, hackathon storyline, and presenting your architecture.

This captures **Phase 1** → **Phase 4** in a clear, professional, step-by-step format.

✅ END–TO–END STEPS (WHAT WE HAVE DONE SO FAR)

Multi-Agent Generative AI System for Automated Regulatory Compliance Reporting

---

🔵 PHASE 1 — DATA & SCHEMA FOUNDATION

Goal: Establish the "ground truth" formats for all downstream agents.

✔ 1. Define Canonical Trade Schema (Internal Standard Format)

We created a unified, clean structure that all raw data will be mapped into.

Example canonical structure:

json

```json
{
  "trade_id": "INT_100000",
  "execution_time": "2024-01-11T23:15:00Z",
  "instrument": {
    "isin": "US7469267437",
    "symbol": "STK029",
    "asset_class": "EQUITY"
  },
  "side": "BUY",
  "quantity": 92,
  "price": 95.0,
  "notional": 8740.0,
  "venue_mic": "XNSE",
  "trader_id": "TRDR260",
  "account_id": "ACC5555",
  "counterparty_lei": "VY01POCGUIRDYNIZBT1G",
  "currency": "USD",
  "order_type": "LIMIT",
  "trading_capacity": "DEAL",
  "short_sell_flag": "N"
}
```

✅ 2. Generate 500 Canonical Trades (Clean Synthetic Data)

This serves as “gold standard input”.

You generated:

✅ `canonical_trades_phase1_500.json`

✅ 3. Define Regulatory Template (RTS22-style)

Created `regulation_template_phase1.json` with:

- Required fields
- Field mappings
- Validation rules

  (Price > 0, Valid ISIN, Valid LEI, etc.)

✅ 4. Create Multiple RAW Data Sources (messy formats)

We generated 10+ realistic raw input examples:

- Broker CSV
- OMS CSV
- EMS JSON
- Exchange batch JSON
- Custodian Nested JSON
- Anomaly-injected CSV

These represent **real-world messy trade feeds**.

🔵 PHASE 2 — INGESTION & NORMALIZATION AGENT

> Goal: Convert **RAW messy inputs → Canonical Trades**

✔️ 1. Created `normalization_agent.py`

This agent:

- Detects schema
- Parses inconsistent timestamps
- Maps columns to canonical schema
- Validates fields (side, qty, price, time, etc.)
- Computes missing fields (notional)

✔️ 2. Normalized RAW CSV + RAW JSON

You ran the agent and got:

```yaml
CSV normalized trades: 25, errors: 5
JSON normalized trades: 5, errors: 2
```

This proves:

- The agent successfully converted valid rows
- Bad rows (invalid quantity, missing price, invalid ISIN, missing LEI) were correctly rejected

✔️ 3. Saved results as:

- `canonical_trades_normalized_phase2.json`

- `normalization_errors_phase2.json`

These represent:

- ✳️ Clean normalized trades
- ❌ Dirty trades (invalid) with error reasons

🔵 PHASE 3 — REGULATION MAPPING & VALIDATION AGENT

> Goal: Take canonical trades and apply regulatory rules.

✔️ 1. Created `regulation_mapping_agent.py`

This agent:

- Loads canonical trades
- Loads the regulation template
- Maps fields:
  - execution_time → TradingDateTime
  - instrument.isin → InstrumentIdentificationCode
  - price → Price
  - quantity → Quantity
  - venue → VenueMIC
  - LEI → BuyerLei
- Validates rules:
  - Price > 0
  - Quantity > 0
  - Valid ISIN
  - Valid LEI

✔️ 2. Ran with 500 clean canonical trades

Output:

```yaml
Loaded 500 canonical trades
Valid regulatory records: 500
```

```
Invalid / failed records: 0
```

Expected — because canonical data is clean.

✅ 3. Produced:

- `regulatory_report_records.json`
- `regulatory_validation_errors.json` (empty for now)

🔵 PHASE 4 — REPORT GENERATION & SUMMARY AGENT

> Goal: Produce regulator-ready reports + human-readable compliance summary.

✅ 1. Created `report_generation_agent.py`

This agent:

- Loads valid + invalid regulatory records
- Builds regulator-facing report (CSV)
- Creates summary JSON
- Creates readable text summary

✅ 2. Outputs:

✅ `regulatory_transaction_report.csv`
Columns include:

- Trade ID
- TradingDateTime
- ISIN
- Price
- Quantity
- VenueMIC
- BuyerLei

✅ `regulatory_report_summary.json`
Contains:

- Total trades

- Valid trades
- Invalid trades
- Validation ratios
- Top error types
- Venue distribution

✅ `regulatory_report_summary.txt`
Human-readable version of above.