DataWeaver - Distributed Agentic Data Pipeline - Capstone Project

Team - 2

Sprint 0

Agenda

- Team Member Roles and Responsibilities
- Problem Statement
- Project Description
- User Personas
- MVP
- Technologies and Algorithms
- Project Schedule - Cadence
- Team Working Agreement
- Sprint 0 Retrospective

Team Members & Roles

Tenzing Salaka

AI Engineer - Engineer Agent

Shreya Anand Kuvalekar

Scrum Master/Tech Lead & System Architect

Orges Velia

AI Engineer-Architect Agent

Priya Prajapati

DevOps & Infrastructure Engineer

Kushwanth Reddy Nomula

Frontend / Dashboard Engineer

Meghana Ventrapragada

Backend & Data Engineer

Sprint 0

Problem Statement

Organizations accumulate fragmented, inconsistent, and poorly documented datasets across departments, resulting in “data swamp” conditions that hinder analytics and decision-making. Integrating such datasets requires significant manual effort to infer schema relationships, identify primary and foreign keys, resolve type mismatches, and validate joins.Existing LLM-based data tools attempt automation but rely on probabilistic reasoning without deterministic safeguards, often hallucinating schema mappings or producing unsafe joins that corrupt downstream analytics. Additionally, most solutions lack distributed scalability and structured separation between semantic reasoning and execution logic.There is a need for a reliable, distributed, self-correcting AI system that separates semantic modeling from deterministic data execution while enforcing invariant-based validation to ensure trustworthy data integration.

Project Description

The Distributed Agentic Data Pipeline is an enterprise-grade AI system that automates dataset mapping, cleaning, and merging using a two-agent architecture. The system decouples semantic reasoning from data execution:

• **Mapper Agent (Architect)** infers schema relationships, identifies keys, and generates ER diagrams.

• **Aggregator Agent (Engineer)** performs schema casting, joins, and deduplication while validating each transformation using deterministic invariants.

The system is deployed across distributed GPU nodes and orchestrated using LangGraph for cyclic self-correction loops. A Streamlit dashboard provides modeling visualization, live validation logs, and downloadable cleaned datasets with an auto-generated data dictionary.

Alex Martinez — Senior Data Engineer
Industry: E-commerce / Healthcare / FinTech
Experience: 5–8 years

Description

It is designed to streamline the integration of fragmented legacy data sources by automating schema interpretation and relationship mapping. It introduces deterministic validation checks—such as row-count reconciliation and null-threshold monitoring—to ensure transformation accuracy. The platform also offers end-to-end transparency into data processing steps and produces downloadable, analysis-ready datasets along with comprehensive data dictionaries.

Goals

- Consolidate multiple unstructured CSV exports originating from legacy systems
- Enable safe, accurate joins while maintaining schema consistency
- Minimize time spent on manual ETL troubleshooting
- Provide dependable, analytics-ready datasets to downstream teams

Challenges

Non-standardized column naming conventions (e.g., *cust_id*, *customerID*, *user_ref*)

- Lack of documentation for legacy data structures
- Join amplification leading to inflated or duplicated row counts
- Undetected schema mismatches corrupting downstream dashboards
- Excessive time spent validating transformations rather than developing pipelines

Priya Sharma — BI Analyst
Industry: Retail / SaaS
Experience: 3–5 years

Description

This system empowers analytics and reporting teams to independently prepare and understand integrated datasets. It provides visual entity-relationship diagrams (ERDs) to clarify how tables connect, along with transparent pipeline logs that document validation steps. Users can download analysis-ready datasets and leverage auto-generated data dictionaries to better interpret column meanings—ensuring confidence in every output produced.

Goals

- Rapidly merge datasets to support reporting needs
- Gain clear visibility into table relationships through ERD views
- Access clean, dependable data without relying on engineering teams
- Build dashboards powered by accurate, trustworthy metrics

Challenges

- Delays caused by dependence on data engineering for cleaned datasets
- Ambiguity around primary and foreign key columns
- Duplicate records distorting KPI and metric calculations
- Insufficient metadata and lack of formal data dictionaries
- Skepticism toward AI-generated joins without validation guarantees

Daniel Kim — Chief Technology Officer / Head of Data

Industry: SaaS / FinTech / Healthcare

Experience: 10+ years

Description

This system is built to deliver enterprise-grade data reliability by combining deterministic validation with scalable AI infrastructure. It introduces invariant-based checks to ensure transformation accuracy, while maintaining full audit trails for governance and compliance. The architecture is designed for distributed, production-ready deployment and automatically generates documentation—including ERDs and data dictionaries—so organizations can trust, verify, and operationalize AI-driven data outputs at scale.

Goals

- Maintain consistent, high-quality data across all departments
- Lower engineering effort required for complex data integrations
- Strengthen governance, traceability, and audit readiness
- Implement scalable, AI-enabled infrastructure
- Showcase innovation through distributed AI system adoption

Challenges

- Data inconsistencies leading to conflicting reports across departments
- Significant engineering costs tied to building and maintaining ETL pipelines
- Limited schema transparency and insufficient documentation
- Concerns around AI systems generating unverified or non-auditable outputs
- Legacy or single-node infrastructure that cannot scale with enterprise demands

MVP — DataWeaver (Refined)

1. CSV Upload & Profiling
Upload multiple CSVs; auto-profile columns, datatypes, sample rows, and row counts — foundational cataloging step.

2. Architect Agent (Mapper)
Analyze headers + samples to infer primary/foreign keys, detect naming inconsistencies, and output validated JSON plus Mermaid ER diagram and standardized data dictionary.

3. Human-in-the-Loop Checkpoint #1
Pause for user confirmation of inferred ERD and mappings before any execution to prevent semantic hallucination.

4. Engineer Agent (Aggregator)
Translate validated mappings into executable Polars/Pandas code, perform schema casting, safe joins, deduplication, and generate clean dataset plus validation report.

5. Deterministic Invariant Enforcement
Enforce row-count invariant $R_{final} \leq R_{source1} * R_{source2}$ during joins; violation halts execution and triggers H-I-T-L review and error logging.

6. Human-in-the-Loop Checkpoint #2
If invariants fail, present generated code and logs for manual correction before re-run to avoid infinite agent loops.

7. Streamlit Dashboard
File upload, ERD visualization, data dictionary, aggregation logs, invariant results, and dataset download UI.

8. Run Logging & Storage
Persist Run ID, architect outputs, engineer outputs, validation results, and status in SQLite/PostgreSQL for auditability.

Sprint 0

Technologies & Algorithms

- **Architecture & Orchestration:** LangGraph — multi-agent cycles with explicit checkpoints
- API Layer: FastAPI for inter-node communication and control

- Networking: Tailscale for secure distributed connectivity

- UI: Streamlit dashboard for visualization & human checkpoints
- **Models:** local gpt-oss:20b (quantized) — Mapper high-reasoning, Aggregator high-throughput

- **Data Processing:** Polars / Pandas (GPU-accelerated when available)

Project Schedule — Cadence

1

Week 0 — Sprint 0 (Discovery)

Requirements, personas, architecture design, and team agreements.

2

Weeks 1–2 — Core MVP Build

CSV profiling, Mapper outputs (ERD + JSON), and H-I-T-L checkpoint #1.

3

Weeks 3–4 — Aggregator & Invariants

Deterministic execution, invariant enforcement, and checkpoint #2 flows.

4

Weeks 5–6 — Dashboard & Logging

Streamlit integration, persistent run logs, dataset downloads, and data dictionary export.

5

Weeks 7–8 — Hardening & Hand-off

Testing, documentation, performance tuning, and demo to stakeholders.

Sprint 0

Team Working Agreement

Team Principles

- Kindness, respect, and patience in all interactions
- Clear ownership and responsibilities
- Even workload distribution; break large tasks
- Document attempted solutions before escalating
- Use GitHub for all updates and reviews

Checklist to Mark Task Complete

- Code runs with no syntax/runtime errors
- Meets acceptance criteria and tests
- Peer review and multi-member verification
- Push to branch and open PR with docs updated
- All relevant documentation tagged and reviewed

Sprint 0 Retrospective

- **What went well**

  Clear problem framing, defined personas, concrete MVP features, and agreed checkpoints to prevent hallucinations.

- **What to improve**

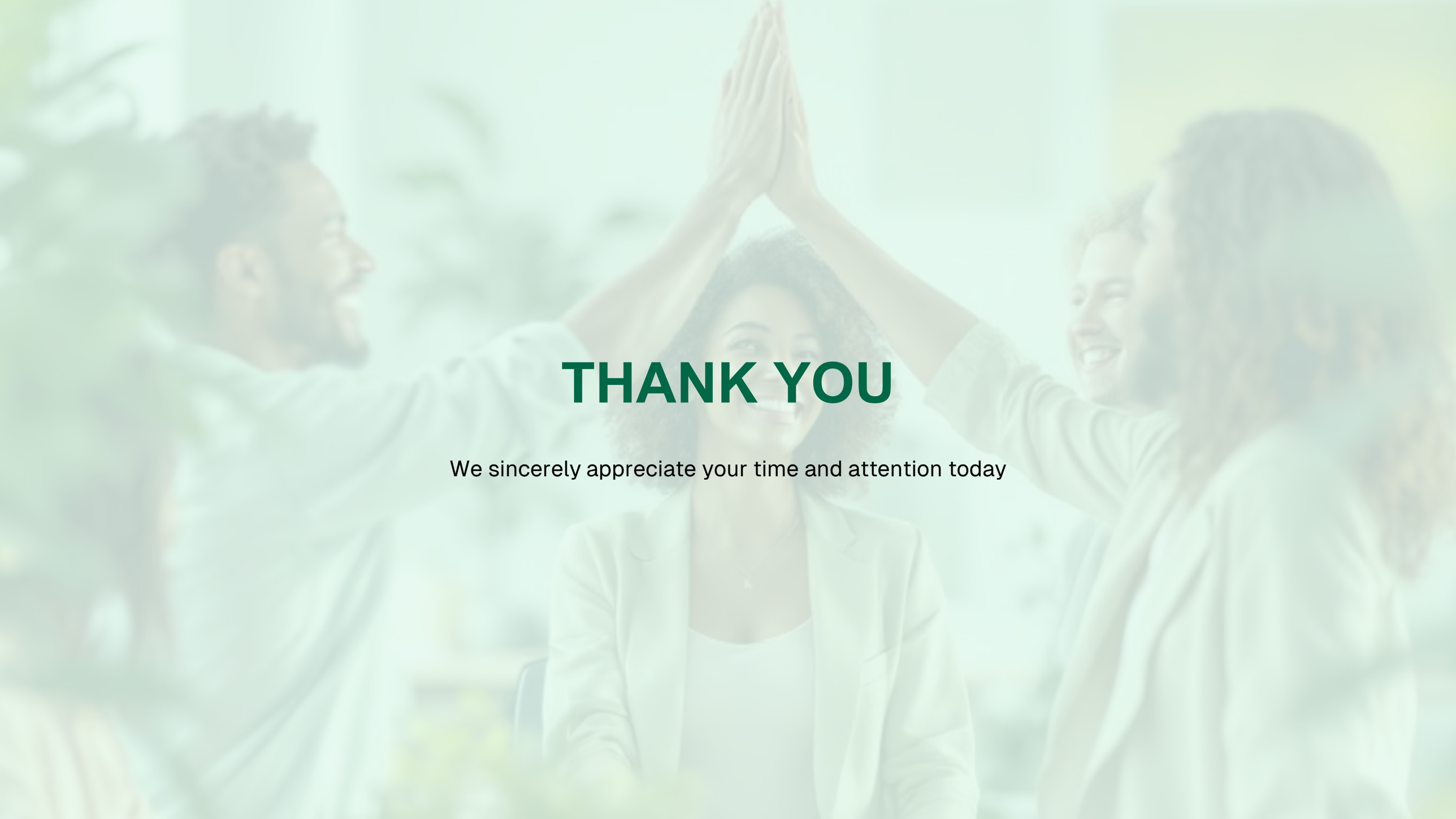
  Prioritize distributed deployment tests, quicker iteration on H-I-T-L UX, and early invariant edge-case simulation.

- **Action items**

  1. Implement CSV profiler + Mapper JSON output (Week 1)
  2. Build H-I-T-L confirmation flow in Streamlit (Week 2)
  3. Implement Aggregator with invariant enforcement and logging (Weeks 3–4)
  4. Persist run metadata in PostgreSQL and finalize dashboard UX (Weeks 5–6)

Wiki Page URL: https://github.com/htmw/S2026A-Team2/wiki

Sprint 0



THANK YOU

We sincerely appreciate your time and attention today