Project Scoping Submission -
LedgerX (AI-Powered Invoice Intelligence Platform with Failure-Aware MLOps)

Team Members:
Jash Bhavesh Shah
Lochan Enugula
Samruddhi Bansod
Rutuja Jadhav
Nirali Hirpara
Deep Bhanushali

1. Introduction

Invoice processing remains one of the most time-consuming and error-prone functions in finance. Organizations depend on Accounts Payable (AP) teams to manually extract information, validate data against purchase orders, map transactions to the general ledger, and reconcile records within ERP systems. This labor-intensive process not only consumes significant resources but also exposes companies to risks of delays, inaccuracies, and compliance issues when managed without intelligent automation.

Current automation (OCR, RPA, ERP add-ons) is limited. OCR is brittle across vendor formats, RPA breaks with schema drift, and ERP add-ons provide no forecasting or anomaly detection. Silent failures (missing totals, mismatched POs) often flow downstream undetected, creating audit risks and financial errors.

LedgerX addresses these gaps with an ML-powered, failure-aware MLOps platform. Unlike commodity tools, it combines:

- Intelligent invoice extraction and standardization into a company schema.
- 3-way validation (Invoice ⇄ Purchase Order ⇄ Receipt).
- Failure-aware prediction to detect errors proactively.
- AI-assisted GL and tax mapping with human-in-the-loop validation.
- Forecasting of cash flow, liabilities, and vendor spend.
- Vendor intelligence graph for anomaly detection and relationship insights.
- Continuous active learning loop to adapt to new workflows.
- Optional chatbot for safe, read-only company queries.
- LedgerX showcases the full ML lifecycle: ingestion, preprocessing, training, deployment, monitoring, drift detection, retraining, and observability.

2. Dataset Introduction

We leverage a combination of public, synthetic, and enterprise datasets to train and evaluate the LedgerX platform's models.

- Public OCR Datasets: SROIE, CORD, DocVQA, and RVL-CDIP — covering diverse invoice, receipt, and document layouts.
- Synthetic Invoices (10,000+): Generated using Python libraries such as Faker, PIL, and ReportLab, incorporating failure scenarios like blurred scans, missing totals, cropped edges, and skewed orientations.
- Enterprise ERP Exports: Anonymized real-world datasets containing invoices, purchase orders, receipts, and general ledger (GL) records.
- APIs: Integrated with QuickBooks Sandbox and Stripe Invoice API for dynamic testing and live inference validation.

Pipeline: -

- Acquisition: Pull from public datasets, ERP exports, synthetic generators.
- Preprocessing: OCR normalization, resize to 300 DPI, blur/contrast detection, schema validation. - Augmentation: skew, noise, missing field injection, rotation, duplicate injection.
- Splitting: stratified by vendor and time period. Dedicated splits for failure prediction and drift detection.

Attribute | **Description**

Dataset Size | 30,000+ documents (≈ 20K real + 10K synthetic)
Storage | ~15 GB
Formats | PDF, PNG, JPEG (documents); JSON/CSV (labels)
Document Types | Invoices (70%), Receipts (20%), Contracts/Forms (10%)
Labels | Vendor, Invoice ID, PO No., Line Items, Totals, Tax, GL Codes, Risk Tags
Languages | Primarily English
Label Quality | Human-verified (real data), programmatic (synthetic data)
Split Ratio | Train 70%
Special Set | Dedicated "failure cases" set labeled for success/failure detection

Data Sources:

- **Public Datasets**: SROIE, CORD, DocVQA, RVL-CDIP
- **Synthetic Data**: Python-generated invoices with randomized vendors, currencies, and formats
- **Enterprise Data**: Anonymized ERP exports from finance systems
- **API Integrations**: QuickBooks Sandbox & Stripe Invoice API for real-time tests

Data Rights & Privacy

- Fully GDPR-compliant handling of sensitive information
- PII anonymization for all vendor and customer identifiers
- Synthetic data is fully owned by the team and used without restrictions
- Audit trails maintained for data access and usage
- Retention policy: Data kept only for the duration of the model lifecycle

Data CharacteristicsInvoice

- Value Distribution: 50% < $5 K | 30% $5 K – $50 K | 20% > $50 K
- Vendor Diversity: 500 + unique vendors (80% invoices from top 50 vendors)
- Label Balance: 85% valid invoices vs. 15% failure cases (intentional imbalance for robust failure detection) Quality Flags:2% blurred images3% missing totals5% schema drift or layout corruption
- Human verification coverage: 100% (real), 95% (synthetic spot-checked)

Management

- DVC + Git LFS for dataset versioning.
- MLflow dataset linkage for reproducibility.
- Great Expectations for schema validation.

3. Data Planning and Splits

Overview

The dataset combines multiple sources — public OCR datasets, anonymized ERP exports, and synthetic invoices — which are processed through a well-defined data pipeline before training. This ensures that all documents are consistent in quality, format, and labeling, while allowing the system to detect and isolate failure cases early in the workflow.

Pre-processing Pipeline

Each document passes through a structured pipeline designed to clean, standardize, and validate the data.

- The steps are as follows:Load the document (PDF or image) and convert it into a standard PIL Image format.
- Detect and correct rotation using the Tesseract orientation API to ensure upright alignment.
- Resize all images to 300 DPI for uniform input quality.
- Convert to grayscale to reduce color noise and improve OCR accuracy.
- Normalize contrast using the CLAHE algorithm to highlight text regions.
- Detect blur using OpenCV Laplacian variance; if the variance is below a fixed threshold, the image is marked as blurred.
- Run OCR normalization with Tesseract and compare the text output with the model's vision-based predictions to check consistency.
- Validate schema using Great Expectations to ensure all mandatory fields (e.g., totals, tax, vendor name) are present and correctly formatted.
- Flag routing:If any issue is detected, the document is moved to the failure prediction set.Otherwise, it is distributed into the training, validation, or testing sets according to the predefined split ratio.

Tools Used

- PIL / Pillow: Image loading and resizing operations
- Tesseract: Optical character recognition and orientation correction
- OpenCV: Blur detection and low-level image analysis
- Great Expectations: Schema validation and data-quality checks
- DVC: Dataset version control and tracking within the MLOps pipeline

4. GitHub Repository Repository:

https://github.com/Lochan9/ledgerx-mlops-final.git

```
project-scoping/
│
├──── README.md                    # Main project introduction, setup, usage
│
├──── data/                        # All datasets used in the project
│    ├──── raw/                    # Original unmodified datasets
│    ├──── processed/              # Cleaned + standardized data
│    ├──── synthetic/              # Synthetic generated data (if any)
│    └──── splits/                 # Train/Val/Test splits
│
├──── src/                         # All source code (model + pipeline)
│    ├──── extraction/             # OCR, parsing, feature extraction
│    ├──── gl_mapper/              # GL code prediction, mapping logic
│    ├──── forecaster/             # Forecasting models (spend, liability)
│    ├──── failure_predictor/      # Quality checks, failure-aware models
│    ├──── training/               # Model training scripts + MLflow hooks
│    └──── monitoring/             # Drift detection, evaluation, alerts
│
├──── inference/                   # FastAPI / REST inference services
│
├──── airflow/                     # Airflow DAGs (preprocessing, retraining)
│
├──── monitoring/                  # Prometheus, Grafana, ELK configs
│                                  # logs, dashboards, drift detectors
│
├──── tests/                       # Unit tests, integration tests, E2E tests
│
├──── docs/                        # Architecture diagrams, scoping file, flowcharts
│    ├──── project_scoping.md      # *Where your entire scoping submission goes*
│    ├──── data_card.md            # Dataset data card
│    ├──── flowchart.png           # Current approach flowchart
│    ├──── bottleneck_analysis.md  # Bottleneck descriptions
│    ├──── metrics_objectives.md   # KPIs & business goals
│    ├──── monitoring_plan.md      # Monitoring plan
│    └──── deployment_architecture.png  # Deployment diagram
│
└──── timeline
```

5. Project Scope:

1. Problems:

- Manual invoice cost and time burden.
- OCR brittleness across vendors.
- Silent failures (bad totals, schema drift).
- Manual GL mapping.
- No forecasting or vendor intelligence.

2. Current Solutions:

- Textract, Google Document AI: OCR only, brittle.
- UiPath, Concur: rule-based, fragile, narrow scope.
- ERP systems: static vendor data, no anomaly detection.

3. Proposed Solutions:

- Extraction & Standardization → into company schema (hybrid ML + fallback rules).
- Validation & Risk Scoring → 3-way match, duplicates, anomalies.
- Failure Prediction → Phase 1 (quality), Phase 2 (semantic).
- AI-assisted GL & Tax Mapping → suggestions validated by humans.
- Forecasting → cash flow, payments using invoices + payment history.
- Vendor Graph → anomaly detection (duplication, spikes).
- Active Learning → corrections feed retraining.
- Optional Chatbot → safe, read-only queries.

6. Current Approach Flow Chart and Bottleneck Detection

1. Current Approach:

Current Approach

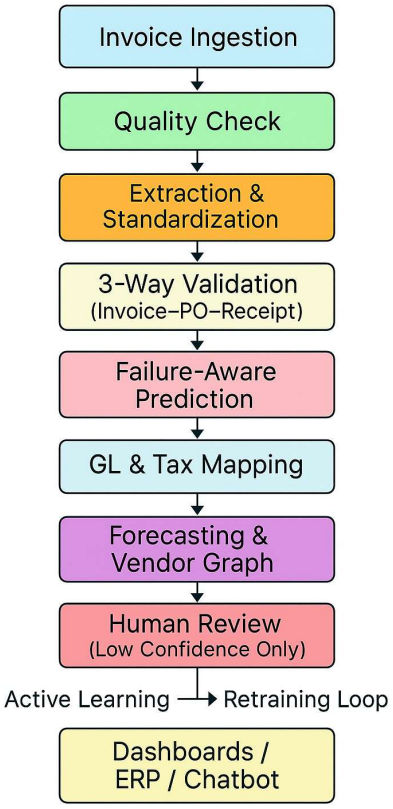
[Invoice Input]
↓
[OCR Extraction]
↓
[Manual Validation]
↓
[ERP Entry]
↓
[Reports / Audit]

2.LedgerX Approach:

Invoice Input
↓
[Quality Check → OCR Normalization]
↓
[ML Extraction & Schema Mapping]
↓
[3-way Validation (Invoice–PO–Receipt)]
↓
[Failure-aware Predictor]
↓
[GL/Tax Mapping + Forecasting]
↓
[Vendor Graph & Anomaly Detection]
↓
[Human-in-the-loop Review]
↓
[Feedback → Active Learning → Retraining]
↓
[ERP Integration / Chatbot]

Invoice Ingestion: Raw PDFs/images collected.

- **Quality Check**: Detect blur, skew, or missing fields before OCR.
- **Extraction & Standardization**: ML-based field extraction normalized to a unified schema.
- **3-Way Validation**: Cross-check invoice–PO–receipt consistency.
- **Failure-Aware Prediction**: Identify potential downstream errors (schema drift, missing totals).
- **GL & Tax Mapping**: Suggest accounting and tax categories via ML models.
- **Forecasting & Vendor Graph**: Predict future liabilities and detect vendor anomalies.
- **Human Review & Feedback**: Only for flagged or low-confidence cases.
- **Dashboards/ERP/Chatbot**: Integrate outputs with finance systems and query interfaces.



1. Invoice Ingestion

All incoming invoices (PDF, PNG, scanned images, emails, uploads, APIs) are collected and sent into the pipeline as unified input documents.

2. Quality Check

Each document is checked for blur, skew, missing fields, and corrupt structure to ensure only valid inputs proceed to extraction; low-quality ones are flagged.

3. Extraction & Standardization

OCR + ML extract key invoice fields and convert all vendor formats into a single standardized company schema (vendor, totals, line items, taxes, etc.).

4. 3-Way Validation (Invoice–PO–Receipt)
The extracted invoice data is cross-validated against purchase orders and goods-receipt records to confirm quantity, pricing, and delivery accuracy.
5. Failure-Aware Prediction
A trained ML model predicts whether the document is likely to fail downstream (e.g., mismatch, missing totals, schema drift) before entering the ERP.
6. GL & Tax Mapping
ML models recommend general ledger codes and tax categories automatically, reducing manual AP workload and improving classification consistency.
7. Forecasting & Vendor Graph
Forecasting models estimate future liabilities/cash flow, while a vendor graph detects anomalies such as duplicate invoices or unusual spending spikes.
8. Human Review (Low Confidence Only)
Only documents with low model confidence or high anomaly risk are routed to a human reviewer for correction, creating a human-in-the-loop process.
9. Active Learning → Retraining Loop
Corrections from humans are fed back into the datasets, and the models are retrained periodically to continuously improve accuracy and robustness.
10. Dashboards / ERP / Chatbot
Processed, validated invoice data is pushed to ERP systems; dashboards display analytics; and a chatbot offers safe, read-only financial queries.

Key Improvements Summary:

Bottleneck	Current Approach	LedgerX Solution	Impact
Wasted Compute	Same model for all docs	Complexity-based routing	65% cost reduction
Silent Failures	No confidence validation	Failure-aware predictor	95% fewer bad outputs
Slow Learning	Quarterly retraining	Continuous active learning	85% → 95% accuracy in 3 months
Poor Review	Random human QA	Targeted review	3x faster model improvement
No Visibility	Black-box OCR	Dashboards + monitoring	<1hr drift detection
Cost-Accuracy Tradeoff	LLMs wasted on simple docs	Tiered inference engine	60–70% infra cost reduction

7. Metrics, Objectives, and Business Goals

ML Metrics:

- Extraction F1 ≥ 90%
- GL Mapping Accuracy ≥ 88%
- Forecasting MAE ≤ 10%
- Failure Prediction AUC ≥ 0.85

Operational Metrics:

- Latency p95 ≤ 5s
- MTTR ≤ 30 minutes
- Drift detection latency ≤ 1h

Business Goals:

- Reduce cost per invoice 60–70%
- Cut manual workload by 70%
- improve audit/compliance readiness
- Deliver CFO-level forecasting intelligence

8. Failure Analysis

Development Risks

Poor data quality, model underperformance, and infrastructure complexity remain the major challenges in this phase.

- **Poor Data Quality**: Variations in vendor formats, blurred images, and missing totals can reduce OCR accuracy. Probability is medium (~35%), and the impact is moderate since noisy data slows training. This is mitigated through synthetic failure datasets, targeted image augmentation, and schema validation using Great Expectations. The data engineering team owns this task, continuing through the first three weeks of preprocessing.

- **Model Underperformance**: Current extraction F1 scores tend to plateau near 82 percent on edge cases such as small invoices and handwritten totals. Probability is medium (~40 percent), but the impact is high because it threatens the MVP target of F1 ≥ 90 percent. Mitigation involves an ensemble strategy combining LayoutLM, EfficientOCR, and a rule-based fallback system. About 2,000 synthetic small invoices are generated with Faker to strengthen coverage. When the ensemble confidence drops below 0.70, the invoice is routed for manual review. The model will be accepted if it reaches F1 ≥ 90 percent on the test set or F1 ≥ 88 percent with less than 2 percent manual review rate. The Data Science and ML teams own this risk, with a checkpoint planned in Week 4.
- **Infrastructure Complexity**: Integrating several components—Airflow, MLflow, Docker, and Weights & Biases—adds operational overhead. Probability is around 30 percent with medium-to-high impact due to potential deployment delays. The MLOps team mitigates this by adopting managed MLflow services, standardized Docker Compose templates, and infrastructure-as-code scripts. Deployment Risks Concept drift, adversarial inputs, and cloud cost overruns can arise after release. Format changes in invoices are handled through drift-detection pipelines and regular retraining. Malformed or intentionally altered invoices are caught by anomaly detection and routed

to manual review. To avoid excessive cost, inference is tiered between lightweight and heavy models, and quantization is used to reduce compute usage.

Deployment Risks

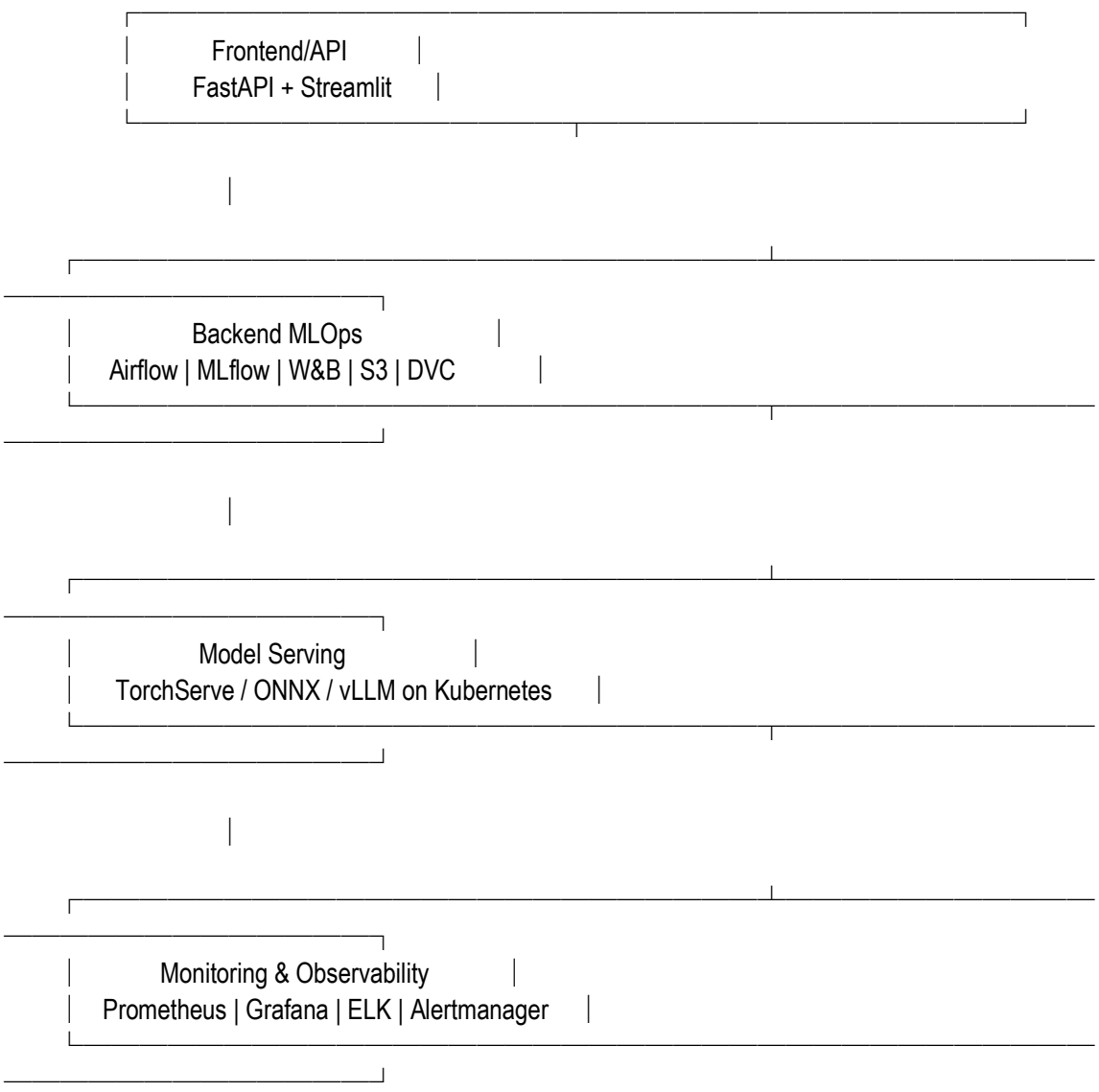
Concept drift, adversarial inputs, and cloud cost overruns can arise after release. Format changes in invoices are handled through drift-detection pipelines and regular retraining. Malformed or intentionally altered invoices are caught by anomaly detection and routed to manual review. To avoid excessive cost, inference is tiered between lightweight and heavy models, and quantization is used to reduce compute usage.

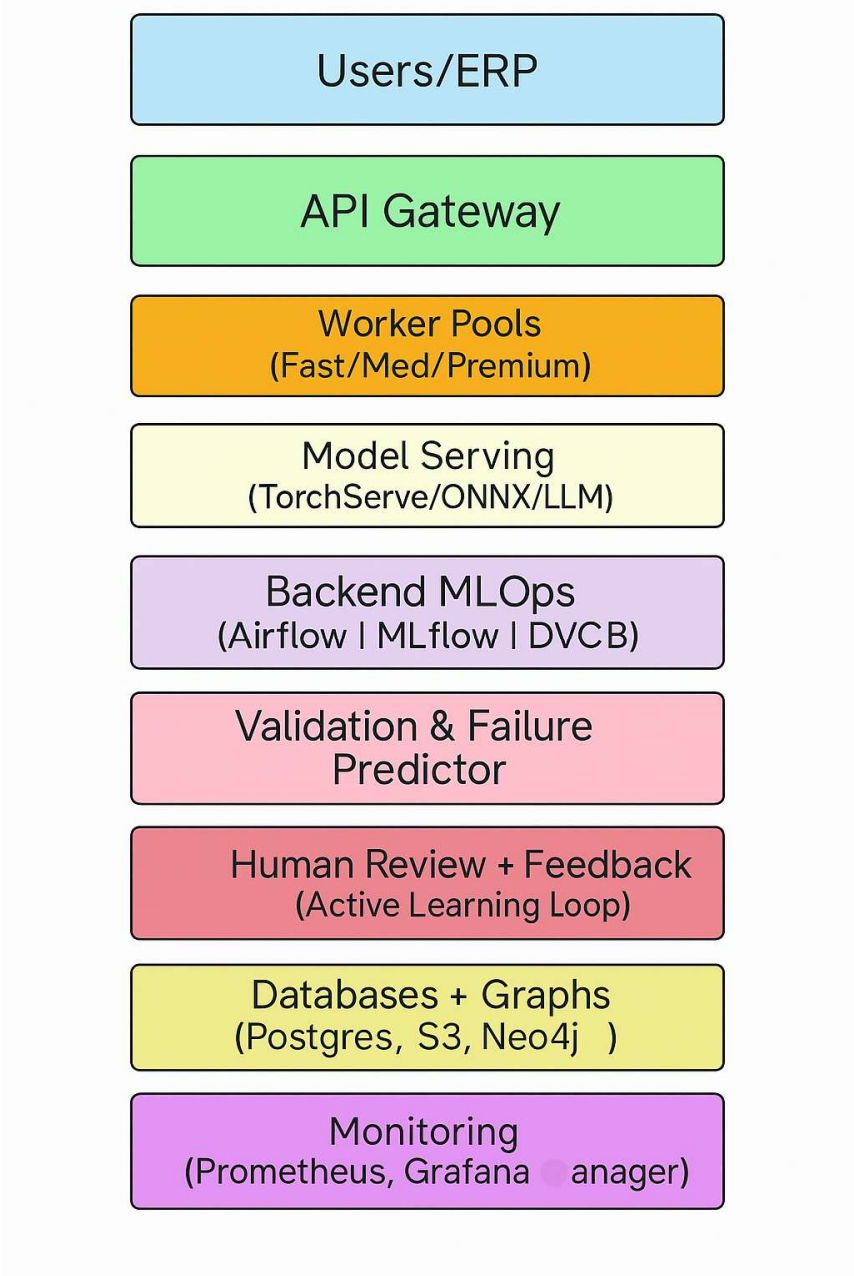
Failure Mode	Impact	Mitigation
Ingestion failures	Data loss or delay	Automatic retries and a Dead Letter Queue (DLQ)
Model-serving crashes	Service downtime	Auto-restart and rollback mechanisms
Training job failures	Interrupted runs	Validation gates and detailed logging
Monitoring blind spots	Missed drift or downtime	Redundant Prometheus and W&B monitoring

9.Deployment Infrastructure

Deployment stack:

- Compute: Kubernetes (EKS/GKE), node pools (CPU, GPU).
- Inference: FastAPI microservices, TorchServe/ONNX/vLLM.
- Data: PostgreSQL (metadata), S3 (storage), Neo4j (vendor graph).
- Pipelines: Airflow for retraining.
- Monitoring: Prometheus, Grafana, ELK, Jaeger, Alertmanager.





Architecture diagram

- Users / ERP: Trigger invoice ingestion or queries via ERP or UI.
- API Gateway: Routes requests to relevant pipelines and manages authentication.
- Worker Pools: Separate queues for low/medium/high complexity documents.
- Model Serving: Deployed with TorchServe / ONNX / vLLM, auto-scaled on Kubernetes.
- Validation & Failure Predictor: ML microservices perform schema checks and risk scoring.
- Human Review + Feedback: Manual verification feeds into the retraining loop.

- Databases + Graphs: PostgreSQL (metadata), S3 (raw data), Neo4j (vendor relationships).
- Monitoring Layer: Prometheus + Grafana for metrics; Alertmanager + Slack for alerts

10.Monitoring & Evaluation

The monitoring system is designed to track the model's real-time performance and ensure consistent accuracy across invoice types, vendors, and data sources. Continuous monitoring helps identify early signs of data drift, pipeline failure, or performance degradation, allowing the team to act before users are affected.

What We Monitor The following key metrics are observed throughout both development and production stages:

- Extraction accuracy and missing field rate — measures how effectively text and key fields are captured. Validation pass/fail ratio — tracks the proportion of documents that meet schema and business-rule checks.
- GL mapping accuracy — evaluates how accurately transactions are matched to general ledger codes.
- API latency (p95) — ensures the response time remains consistent for end-users.
- Forecast mean absolute error (MAE) — measures prediction accuracy in financial forecasting modules.
- Data drift signal (Kolmogorov-Smirnov test) — detects distribution shifts in incoming invoices.

Metric	Normal Range	Warning	Critical	Action
Extraction F1 Score	≥ 90%	85–90%	< 85%	Send Slack alert and trigger the retraining DAG
Validation Pass Rate	≥ 92%	88–92%	< 88%	PagerDuty notifies on-call engineer; review last 100 invoices
GL Mapping Accuracy	≥ 88%	85–88%	< 85%	Email ML team; disable GL suggestions if accuracy < 80%
API Latency (p95)	≤ 5 s	5–8 s	> 8 s	Inspect Kubernetes logs; scale pods if CPU > 80%
Forecast MAE	≤ 10%	10–12%	> 12%	Check for data drift; retrain if vendor formats changed
Drift Signal (KS Test)	p > 0.05	0.02–0.05	p < 0.02	Trigger drift alert; pause collection; investigate root cause

11. Success and Acceptance Criteria

MVP Success Criteria (Go/No-Go at Week 6)

The Minimum Viable Product (MVP) represents the first fully functional version of the LedgerX platform. By the end of Week 6, the team must demonstrate a stable OCR extraction pipeline, operational three-way validation, initial failure-aware detection, and accurate GL mapping suggestions. The following measurable

targets define whether the MVP is considered a Go or No Go for progression to the next development phase.

Area	Success Indicator	Acceptance Threshold
OCR Extraction Pipelin	End-to-end processing of 1,000 test invoices without runtime errors	Successful run on all test invoices
Extraction Accuracy	Field-level performance	F1 ≥ 85% (Precision ≥ 85%, Recall ≥ 85%)
3-Way Validation	Invoice–PO–Receipt matching speed and accuracy	90% of invoices matched within 5 minutes
Failure Detection (Phase 1)	Detection of blur or missing-field cases	Recall ≥ 95% on labeled failure set
GL Mapping Suggestions	Human validation of suggested GL codes	≥ 80% acceptance rate by reviewers
API Latency (p95)	System responsiveness	≤ 8 seconds
Testing Coverage	Code quality and maintainability	Unit ≥ 80%, Integration ≥ 60%
Documentation	Repository completeness	README includes setup, usage, and API documentation

No-Go Conditions

- The MVP will be delayed or re-scoped if any of the following occur:
- Extraction F1 < 80 % → Reallocate Weeks 7–8 for retraining and data cleanup
- API latency > 15 s → Investigate performance bottlenecks and optimize endpoints
- 3-Way validation fails on > 10 % of invoices → Review and debug validation logic and schema rules

12. Timeline Planning

Week 1–2: OCR & Extraction

Objective: Establish a reliable OCR and extraction baseline capable of processing both real and synthetic invoices.

Day	Task	Details
Day 1–2	Set up LayoutLM baseline on SROIE	Benchmark initial F1 score for field-level extraction
Day 3–4	Fine-tune on 5,000 real invoices	Evaluate F1 score on a held-out set of 500 samples
Day 5	Implement rule-based fallback	Handle malformed or partially scanned PDFs
Day 6	Build DVC pipeline	Automate ingestion and versioning for 10,000 synthetic invoices

Day	Task	Details
Day 1–3	Implement three-way matching logic	Validate invoice, PO, and receipt alignment with unit tests
Day 4	Add duplicate detection	Use Jaccard similarity on extracted text fields
Day 5	Design rule-based anomaly scoring	Detect outliers in line-item and total amounts
Day 6–7	Perform integration testing	Test full flow; resolve schema mismatches

13. Additional Information Details

- We plan to design the project with flexibility in mind, enabling streamlined additions within the current schedule and future development needs
- Differentiator: Failure-aware MLOps, not just OCR.
- Scalability: 1M+ invoices/month.
- Business Impact: CFO-level insights, reduced costs.
- Innovation: Integrates extraction, forecasting, anomaly detection, continuous learning, and monitoring in one system.