Project Scoping Submission: FinSight

Project: FinTech Insights - A financial intelligence platform for SEC Filings.

Team Members:

1] Nicholas Nehemia
2] Sridipta Roy
3] Syeda Tooba Ali
4] Karthik Raja Subramanian
5] Vishak Nair
6] Joel Markapudi

1. Introduction

Financial analysts, Investors, and portfolio managers currently spend an excessive amount of time - often hours - manually parsing hundreds of pages in 10-K annual filings. The annual report on Form 10-K provides a comprehensive overview of the company's business and financial condition and includes audited financial statements. This process is not only inefficient but also prone to human error and lacks a systematic method for auditing or tracing data back to its source. The objective is to find specific Key Performance Indicators (KPIs), understand the management narrative behind them, and verify their context, which is a significant operational bottleneck.

In this project, we will present **an AI-powered financial intelligence platform** that extracts key metrics, generates evidence-backed narrative summaries, and highlights risk factors from 10-K filings. The solution will have:

1. **A Structured KPI Extractor:** A robust pipeline that identifies, parses, normalizes (units, currencies, periods), and validates numerical KPIs from filings, outputting them in a structured, machine-readable format (JSON) with direct citations.
2. **An Explanation Engine:** A Retrieval-Augmented Generation (RAG) system that finds relevant passages to answer qualitative "why" questions, providing concise, evidence-based summaries with citations to the source text.

The presented output would provide auditable key metrics, evidence-backing narrative summaries, and potentially summarized risk factors, investment guidelines, or market reaction correlation.

2. Dataset Information

2.1 Dataset Introduction

The dataset we decided to pick as our core resource for this project is essentially a well-prepared version of the SEC 10-filings: "khaihernlow/financial-reports-sec". We also plan to include the potential of live-data feeds realized, such as parsed-API data into our data engineering pipeline.

This dataset provides a comprehensive collection of US-GAAP financial data extracted from the financial statements of U.S. exchange-listed companies. The data is sourced from filings submitted to the U.S. Securities and Exchange Commission (SEC) via the EDGAR (Electronic Data Gathering, Analysis, and Retrieval) database, covering the period from January 2009 onwards.

The purpose of this dataset is to offer a structured and accessible way to analyze the financial health and performance of public companies in the United States. Its relevance lies in its potential for financial analysis, academic research, and machine learning applications related to corporate finance and investment. The dataset focuses on providing key financial figures as reported by the companies, ensuring a direct and unaltered reflection of their financial statements.

2.3 Data Card

- **Name:** financial-reports-sec
- **Source:** SEC EDGAR 10-K filings
- **Granularity:** Sentence-level
- **Format:** JSONL
- **Size:** Multiple configurations (small/large, lite/full). Example: small_full contains ~240k sentences.

Feature	**Description**
Granularity	Each row = one sentence from a 10-K filing. Rows include both textual content (sentence) and metadata (e.g., docID).
Content	Management commentary (MD&A), financial statements, risk disclosures.
Size	200-400 MB on smaller sets, **10–12 GB** on large full, ler sets,
Data Types	Strings, Categorical/Numeric, Composite Fields, Float Fields.
Use cases	Pretraining/finetuning, Sentiment Analysis, Market Reaction Predictions, Risk Summarization, narrative QA.

The primary source of the data is the **U.S. Securities and Exchange Commission (SEC)**. Specifically, the data is extracted from the following sources:

- **SEC EDGAR Database:** The central repository for all public company filings.
- **SEC Financial Statement Data Sets:** Curated datasets provided by the SEC, which can be accessed at https://www.sec.gov/data/financial-statements.
- **EDGAR Application Programming Interfaces (APIs):** The SEC provides APIs for programmatic access to EDGAR data.

2.3 Data Sources

The dataset is hosted on HuggingFace Datasets, originally compiled from SEC EDGAR filings. It leverages JSONL files of parsed text segments aligned with filing metadata. The HuggingFace page provides both raw data access and documentation:
- https://huggingface.co/datasets/khaihernlow/financial-reports-sec

2.4 Data Rights and Privacy

The dataset is composed of publicly available SEC filings. Since 10-K reports are mandatory disclosures for public companies, there are no personal privacy risks or GDPR implications. The SEC provides a disclaimer that the data is "as filed" by the companies and does not guarantee its accuracy. Users of the dataset should be aware that there may be occasional errors or inconsistencies in the reported financial figures. The license is described as "us-pd" (U.S. Public Domain), and more information can be found at http://www.usa.gov/publicdomain/label/1.0/. Licensing is aligned with open research and non-commercial analysis, and the HuggingFace dataset page notes its intended use for NLP, information retrieval, and financial AI tasks.

3. Data Planning and Splits

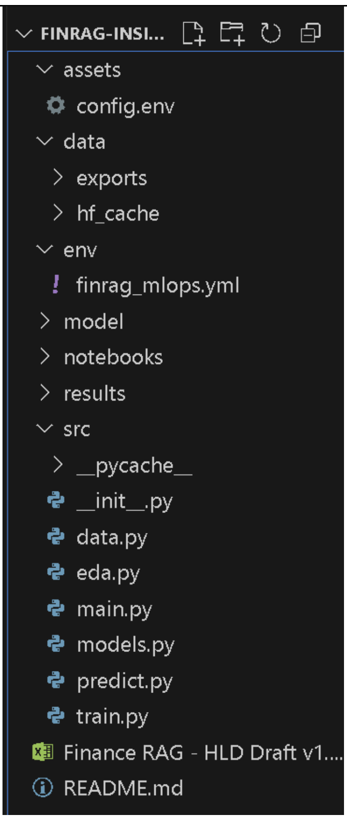
For experimentation, we will use the **small_full configuration**, which provides the complete set of features (including labels and returns) at a manageable scale. Data preprocessing includes:

Loading & Persistence: Data is downloaded once via HuggingFace datasets API, then cached locally for stable offline work.

- **Sampling:** For development, we plan to subsample ~300 sentences for lightweight EDA and pipeline testing. We might use more, depending on context/algorithm limits.
- **Splits:** The dataset already provides train, validation, and test splits. We will use these directly to maintain reproducibility, while allowing smaller random samples for iterative development.
- **Preprocessing:** Key steps might include:
    - Normalizing units and periods from text for KPI extraction.
    - Sentence windowing (2–3 sentences) to form retrieval chunks.
    - Adding derived metadata fields (e.g., normalized fiscal year, standardized KPI names).

4. GitHub Repository

The GitHub repository is: https://github.com/mjsushanth/finrag-insights-mlops

This is a screenshot of our initial project structure: We would depend on yml files for local packaging of environments cleanly using conda. Environment parameterization will be done with the help or env or json files.	

```
FINRAG-INSI...
  assets
    config.env
  data
    exports
    hf_cache
  env
    finrag_mlops.yml
  model
  notebooks
  results
  src
    __pycache__
    __init__.py
    data.py
    eda.py
    main.py
    models.py
    predict.py
    train.py
  Finance RAG - HLD Draft v1....
  README.md
```

5. Project Scope

5.1 Problems

- Manual review of 10-K filings is time-intensive: analysts must sift through hundreds of pages to find KPIs like revenue or R&D.
- Inconsistent formats, fiscal periods complicate KPI extraction and comparison — metrics appear in different units (millions, billions), across different fiscal year-ends, and may be repeated in multiple sections.
- Cross-company taxonomy mismatches and lack of context limit analysis.
- Risk factors (Item 1A) are verbose, repetitive, and hard to distill into insights.
- Market reaction to filings is not easily traceable back to language in reports.
- Current solutions (dashboards like Bloomberg/FactSet) are proprietary and not easily customizable or auditable.

5.2 Current Solutions

- Traditional BI dashboards (Power BI, Tableau) rely on structured databases, not unstructured text.
- NLP research has focused on financial sentiment classification or document-level embeddings, but few solutions integrate structured KPI extraction with evidence.
- SEC itself provides XBRL data, but it lacks narrative context and often misses subtleties of management commentary.

5.3 Proposed Solutions

Our system introduces a dual-layer RAG framework:

- Structured RAG for extracting KPIs (Revenue, Net Income, R&D, EPS) with unit/period normalization and explicit evidence citations.
- Classic RAG for summarizing and highlighting key risk factors and management commentary, with strict citation rules.
- Evidence-based reporting: results are auditable (linked to exact sentences/sections) and combine metrics + narrative in a single fused report.

Deliverable: A prototype dashboard/report generator that outputs a summary per filing with KPI tables + explanatory notes. Potentially more, with risk narratives or market reactions.

6. Current Approach and Bottleneck Detection

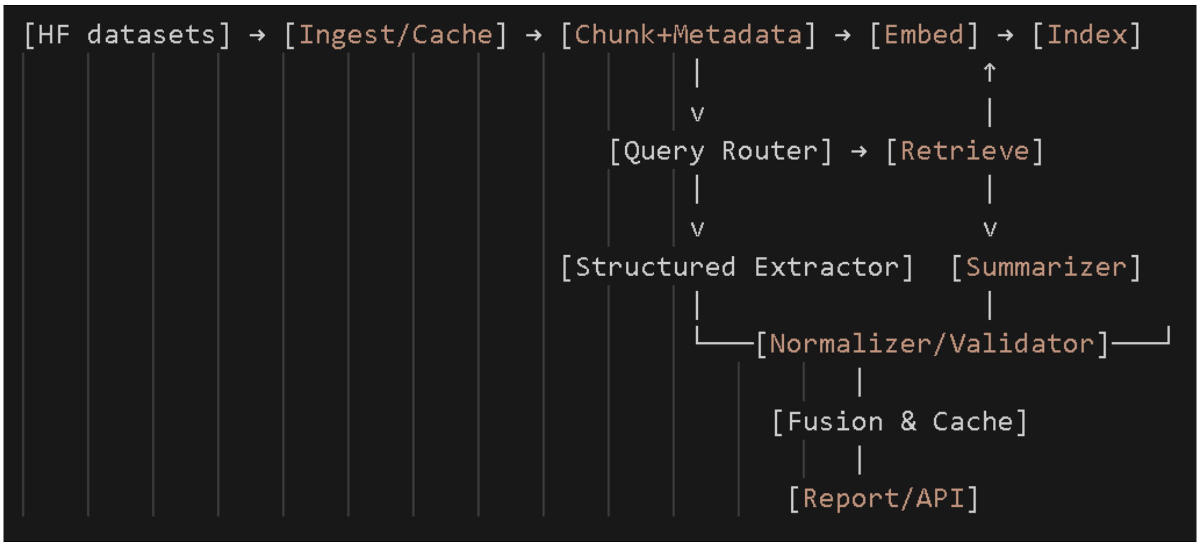
6.1 Current manual process (what analysts do today)

1. Pull a 10-K from EDGAR → scan Item 7 (MD&A), Item 8 (Financial Statements), Item 1A (Risk Factors), etc.
2. Copy metrics (Revenue, Net Income, R&D, Operating Income, EPS) into sheets; normalize units (“in millions/billions”).
3. Align periods (different fiscal year-ends), dedupe repeated mentions (tables vs prose), and add notes explaining changes. Build a deck/dashboard.

6.2 High-Level Architecture (AI-assisted, aligned to the dataset)

Core components:

- **Ingestion** (HF → Parquet snapshot): reproducible local cache.
- **Preprocess & Chunking**: window 2–n consecutive *sentences*; attach rich metadata.
- **Embedding & Index**: sentence-transformer embeddings in FAISS (dev) or Qdrant (scale).
- **Retrieval**: company/period filters + section priors → ANN search → top-k passages.
- **Structured Extractor**: LLM as a **constrained parser** (KPI value, unit, , citations).
- **Normalizer & Validator**: deterministic unit/period normalization + sanity checks.
- **Narrative Summarizer**: short, citation-bound bullets from MD&A / Risk Factors.
- **Fusion & Cache**: assemble KPI table + narrative; cache (cik, period, kpi) results.
- **Observability**: metrics, logs.



```
[HF datasets] → [Ingest/Cache] → [Chunk+Metadata] → [Embed] → [Index]
                                          |                  ↑
                                          v                  |
                                     [Query Router] → [Retrieve]
                                          |                  |
                                          v                  v
                                  [Structured Extractor]  [Summarizer]
                                          |                  |
                                          └───[Normalizer/Validator]───┘
                                                    |
                                             [Fusion & Cache]
                                                    |
                                              [Report/API]
```

(Quick flowchart on the process flow. Potential execution modules might look like this.)

The intermediate – coding outputs might be objects such as: KPI cache row, Report payload, or Passage records.

6.3 Bottlenecks

Ingestion issues overwhelm RAM/IO, **Compute time** & index size, Hallucinations**,** Unverifiable claims during summarization, Retrieval issues such as wrong section, missed scale cues. Also, we could have model drift and quality changes after certain tweaks.

Some bottlenecks are predictable (embedding, retrieval precision, unit/period ambiguity, LLM latency) and we attempt with simple controls at each stage to keep the system stable as it grows.

7. Metrics, Objectives, and Business Goals

➢ **Business Goal:** Building an automated, AI-powered system that reduces the manual KPI gathering + note-taking time, that provides Auditability & trust, and which helps with faster onboarding. This system should also have the potential to scale to near real-time needs.

7.1 Objectives (what "success" means)

- **O1 — Accurate KPIs with evidence:** Extract a small set of headline metrics from 10-K filings with **auditable citations**. The number of metrics extracted concretely could vary.
- **O2 — Useful explanations:** Provide short, **evidence-backed** narratives (Item 7/1A) that help an analyst understand drivers/risks.
- **O3 — Operational readiness:** Deliver results **fast** and **consistently** (cacheable, low latency, predictable cost). Establish a fully functional MLOps pipeline that can deploy a model update (retrained or newly developed).
- **O4 – Automation:** Achieve ≥80% automated data extraction and structuring from new 10-K filings (reducing manual data prep time to zero).

7.2 Metrics

Category	What to Measure	Key Focus
A) Retrieval Quality	Whether the system brings back the right passages from filings.	Recall (did we capture the relevant section?) and ranking quality (best matches near the top).
B) Structured KPI Extraction	Accuracy of extracted numbers, units, and periods after normalization.	Correct values, valid evidence citations, and coverage (how often a KPI is returned confidently).
C) Narrative Summaries	Quality and reliability of generated explanations.	Faithfulness to evidence, usefulness/clarity via human review, and no invented numbers.
D) System & Operations	Overall performance and consistency of the pipeline.	Response time per query, stability of the pipeline, and consistency of repeated queries.

How to measure:

We Build a tiny, reliable **gold set** (once). Scope: 15–25 filings, 3–5 industries, 2 years each if possible. We extract KPIs, possibly Revenue, Net Income, R&D, Operating Income. We extract a solid ‘evidence narrative’, ‘normalized value’, and docID, section, sentence indices. It is a short guideline. That gold set powers all the measurements below.

- **Retrieval Quality:** Score ranking quality; human scoring or judge-LLM evaluation.
- **Exact Match (EM) on value:** Compare extracted `value_normalized` to gold after unit scaling; allow a small tolerance.
- **Coverage:** % of (company, period, KPI) where the system returns an answer.
- **Faithfulness, Citation check:** Verify each claim is supported by cited passages.
- **Human Utility (score):** Two reviewers score 1–5 on clarity and usefulness.
- **Latency per report:** time from request → KPI+summary. Track p50/p95.
- **Stability:** error rate and auto-retry success.

8. Failure Analysis

LLM Responsiveness and Performance

- *Risk:* The LLM may become slow or unresponsive under high demand.
- *Mitigation:*
  - Use monitoring tools to track response time and availability.
  - Scale resources dynamically during high-load periods.

LLM Hallucinations (Numbers & Statements)

- *Risk:* LLMs may hallucinate, especially with financial figures, producing numbers or claims not found in the filings.
- *Mitigation:*
    - Require all outputs (numbers and statements) to be traceable to specific evidence.
    - Automatically discard any output that lacks supporting citations.

Data Spikes During SEC Filing Days

- *Risk:* Release days involve hundreds of filings, creating spikes in data volume and processing load.
- *Mitigation:*
    - Implement a caching layer to store processed KPIs and summaries.
    - Serve repeat queries from cache rather than re-processing through the LLM pipeline.
    - Use asynchronous pre-processing to handle spikes in workload.

ETL Pipeline Reliability

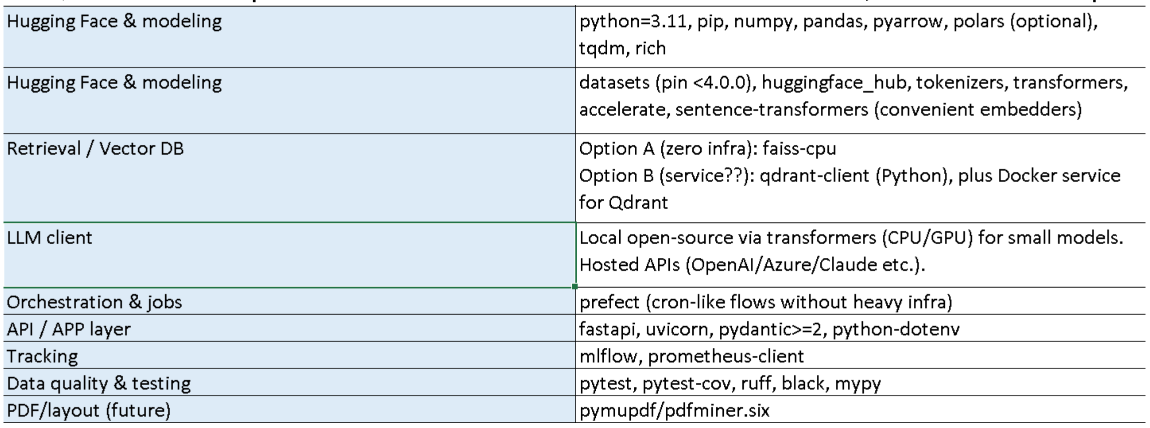
- *Risk:* The ETL pipeline during SEC filings release may fail or miss data.
- *Mitigation:*
    - Continuously monitor the data ingestion pipeline.
    - Add alerts for ingestion failures or schema mismatches.

Possible Evolving Nature of 10-K Filings

- *Risk:* The structure of 10-K filings is not static; the SEC may change requirements.
- *Mitigation:*
    - Incorporate continuous monitoring for **data drift** and **concept drift**.
    - Establish a feedback loop where analysts can flag incorrect extractions.

9. Deployment Infrastructure

First, this is the plan we have **for the local infrastructure,** for the development process.



Hugging Face & modeling	python=3.11, pip, numpy, pandas, pyarrow, polars (optional), tqdm, rich
Hugging Face & modeling	datasets (pin <4.0.0), huggingface_hub, tokenizers, transformers, accelerate, sentence-transformers (convenient embedders)
Retrieval / Vector DB	Option A (zero infra): faiss-cpu Option B (service??): qdrant-client (Python), plus Docker service for Qdrant
LLM client	Local open-source via transformers (CPU/GPU) for small models. Hosted APIs (OpenAI/Azure/Claude etc.).
Orchestration & jobs	prefect (cron-like flows without heavy infra)
API / APP layer	fastapi, uvicorn, pydantic>=2, python-dotenv
Tracking	mlflow, prometheus-client
Data quality & testing	pytest, pytest-cov, ruff, black, mypy
PDF/layout (future)	pymupdf/pdfminer.six

We also have a environment package details file (yml) file which can be easily managed through mamba/conda, in the github repository. It is the current envrionment file and gets updated over time.

Cloud Deployment Infrastructure:

Layer	Services	Why (Rationale)
Dev & ML Scripting	**SageMaker Studio / Notebooks**	Managed workspace for notebooks and Python; easy scaling to jobs.
Pipelines / Orchestration	**SageMaker Pipelines**	First-class ML DAGs; reproducible, trackable steps end-to-end.
Artifact & Data Storage	**Amazon S3**, **Amazon ECR**	Durable data/model storage; container registry for jobs and serving.
Vector Store (RAG)	**Amazon OpenSearch Serverless (k-NN)**	Managed, scalable vector search with minimal ops.
Embeddings	**Amazon Bedrock (Titan Embeddings)**	Serverless embeddings; simple integration and billing.
LLM Inference (RAG)	**Amazon Bedrock (Claude/Llama/Titan)**	Reliable, governed LLM access; no infra to manage.
KPI Cache / Feature Store	**Amazon DynamoDB**	Low-latency key–value store for `(cik, period, kpi)` results.
API / Serving	**AWS Lambda + API Gateway**	Serverless HTTPS endpoint; auto-scale; minimal ops.
Front-End / UI	**AWS Amplify (React app)**	Fast, managed hosting for the analyst dashboard.
Monitoring & Logging	**Amazon CloudWatch**, **SageMaker Model Monitor**	Centralized logs/alerts; drift/quality checks on schedule.
Secrets & Config	**AWS Secrets Manager / SSM Parameter Store**	Secure, versioned configuration and credentials.
CI/CD	**GitHub Actions → CodeBuild/CodePipeline**	Automate build, test, deploy for images, pipelines, and API.
Live EDGAR Ingestion (Streaming)	**AWS Lambda → Amazon SQS → Batch Workers (SageMaker Processing or Lambda)**	Event-driven intake; queue smoothing; scalable async processing into S3/OpenSearch/DynamoDB.

10. Monitoring Plan

Once deployed, we aim around Data quality monitoring, Pipeline monitoring, Business monitoring, Alerts and logging.

- Extraction metrics: track Exact Match (values), unit accuracy, evidence/claim precision.
- Summarization metrics: track citation coverage (claims supported by evidence).

- Collect p50/p95 latency for retrieval and generation, and error rate of failed queries.
- Structured logs store (cik, reportDate, kpi, value, evidence, confidence).
- Monitoring dashboards (Grafana/Prometheus optional).
- Potentially Track dataset drift (new companies, missing sections, unusual sentence lengths).
- Monitor quality to assess for hallucination.

This continuous monitoring approach will allow us to maintain the platform's effectiveness while building trust with users who rely on accurate, auditable financial insights for their decisions.

11. Success and Acceptance Criteria

Technical Success:

- The system can ingest and process the dataset reliably.
- KPI extraction produces accurate numbers with correct units, traceable evidence.
- Narrative summaries are supported by cited text and free of invented numbers.
- Pipeline runs consistently with acceptable latency.

Business/Analyst Acceptance:

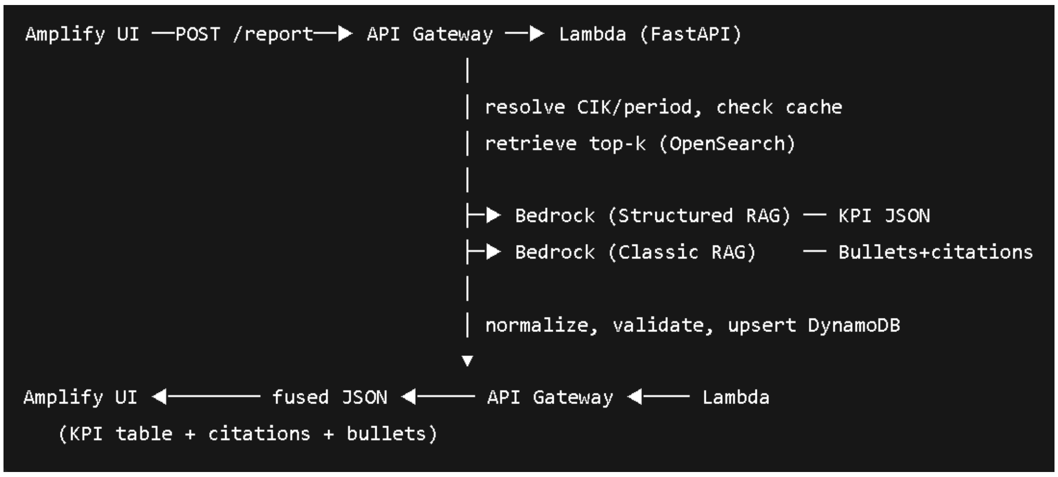
- The fused report (KPIs + narratives) reduces manual review effort, its is clear to interpret.
- Users trust the results and evidence, or the finQA part.
- The system demonstrates a good solution and scalability.

12. Timeline Planning

Week 1: Scoping & Requirements Gathering.
Weeks 2–4: Baseline Model & Initial Data Architecture.
Weeks 5–6: Scaling & Data Pipeline.
Week 7: Deployment & Monitoring.
Week 8: Buffer & Final Review.

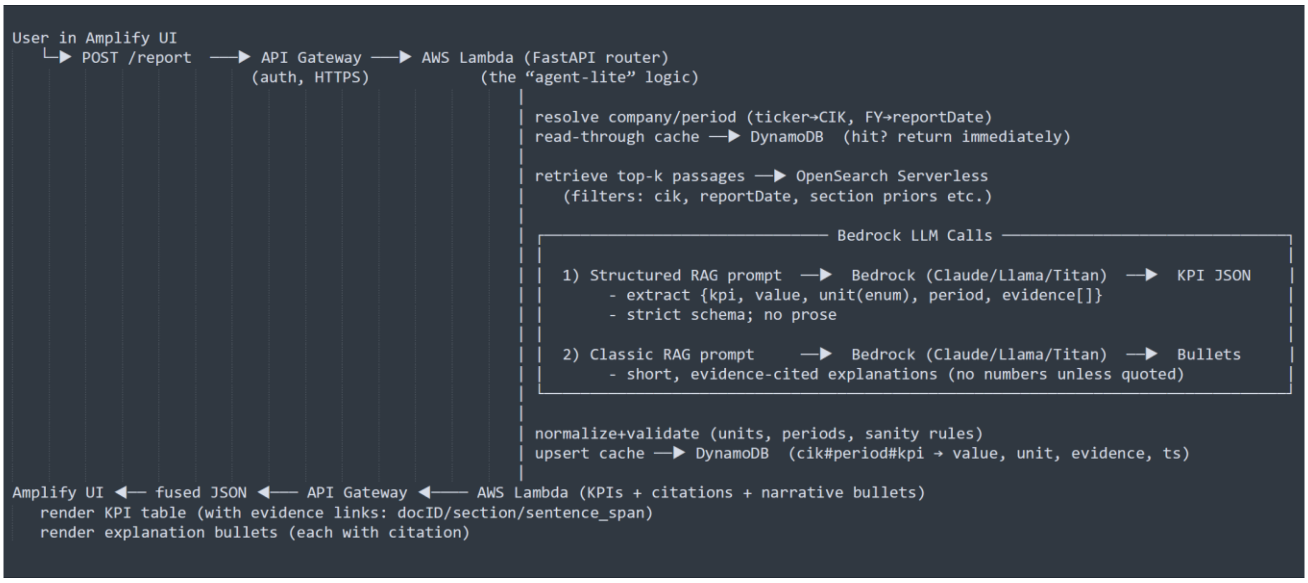
The project will follow an eight-week phased timeline to balance research, implementation, and delivery.

13. Additional Information



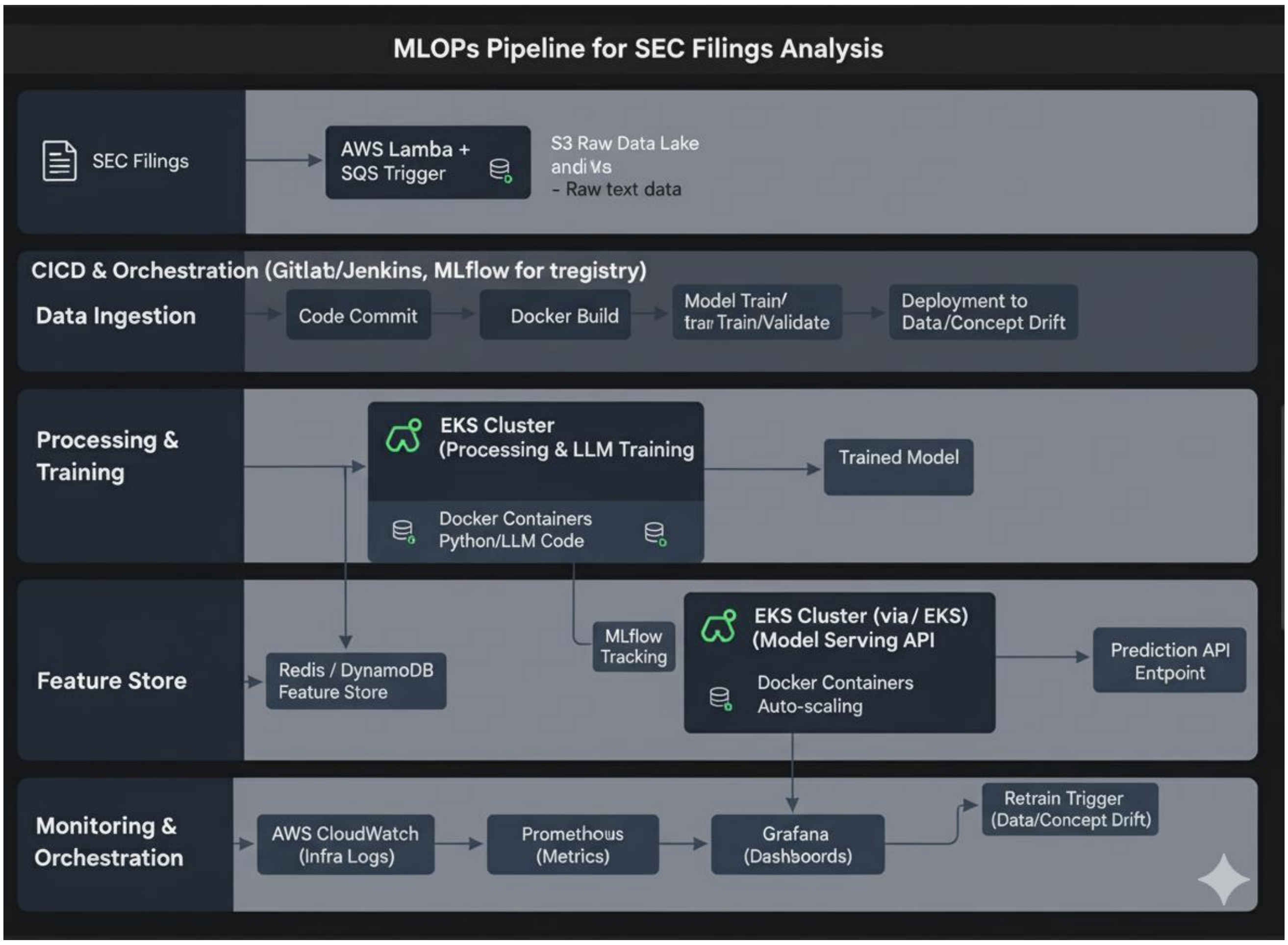
```
Amplify UI ──POST /report──▶ API Gateway ──▶ Lambda (FastAPI)
                                             │
                                             │ resolve CIK/period, check cache
                                             │ retrieve top-k (OpenSearch)
                                             │
                                             ├─▶ Bedrock (Structured RAG) ── KPI JSON
                                             ├─▶ Bedrock (Classic RAG)    ── Bullets+citations
                                             │
                                             │ normalize, validate, upsert DynamoDB
                                             ▼
Amplify UI ◀──────── fused JSON ◀───── API Gateway ◀───── Lambda
   (KPI table + citations + bullets)
```

(Quick Sequence Diagram on User Request/Query - flow.)



```
User in Amplify UI
   └─▶ POST /report  ──▶ API Gateway ──▶ AWS Lambda (FastAPI router)
                        (auth, HTTPS)           (the "agent-lite" logic)
                                                  │
                                                  │ resolve company/period (ticker→CIK, FY→reportDate)
                                                  │ read-through cache ──▶ DynamoDB  (hit? return immediately)
                                                  │
                                                  │ retrieve top-k passages ──▶ OpenSearch Serverless
                                                  │   (filters: cik, reportDate, section priors etc.)
                                                  │
                                                  │ ┌──────────────────────── Bedrock LLM Calls ─────────────────────────┐
                                                  │ │                                                                    │
                                                  │ │  1) Structured RAG prompt  ──▶  Bedrock (Claude/Llama/Titan)  ──▶  KPI JSON  │
                                                  │ │       - extract {kpi, value, unit(enum), period, evidence[]}       │
                                                  │ │       - strict schema; no prose                                    │
                                                  │ │                                                                    │
                                                  │ │  2) Classic RAG prompt     ──▶  Bedrock (Claude/Llama/Titan)  ──▶  Bullets   │
                                                  │ │       - short, evidence-cited explanations (no numbers unless quoted)│
                                                  │ └────────────────────────────────────────────────────────────────────┘
                                                  │
                                                  │ normalize+validate (units, periods, sanity rules)
                                                  │ upsert cache ──▶ DynamoDB  (cik#period#kpi → value, unit, evidence, ts)
                                                  │
Amplify UI ◀── fused JSON ◀── API Gateway ◀─── AWS Lambda (KPIs + citations + narrative bullets)
   render KPI table (with evidence links: docID/section/sentence_span)
   render explanation bullets (each with citation)
```

(Cloud – Infrastructure diagram that supports section 9 (deployment infra). This might be subject to change or improvements.)


MLOPs Pipeline for SEC Filings Analysis
SEC Filings
AWS Lamba + SQS Trigger
S3 Raw Data Lake andi Vs - Raw text data
CICD & Orchestration (Gitlab/Jenkins, MLflow for tregistry)
Data Ingestion
Code Commit
Docker Build
Model Train/ trar Train/Validate
Deployment to Data/Concept Drift
Processing & Training
EKS Cluster (Processing & LLM Training
Docker Containers Python/LLM Code
Trained Model
Feature Store
Redis / DynamoDB Feature Store
MLflow Tracking
EKS Cluster (via / EKS) (Model Serving API
Docker Containers Auto-scaling
Prediction API Entpoint
Monitoring & Orchestration
AWS CloudWatch (Infra Logs)
Promethous (Metrics)
Grafana (Dashboords)
Retrain Trigger (Data/Concept Drift)
Image Credits on 3 – (created using textflows on notepads / draw, then enhanced with google AI creation.)