Retail Stock AI Pipeline — System Design & Recommendations

Created: 2025-11-19

Executive summary

This document describes a fully cloud-based hybrid architecture for a retail-stock AI assistant used by **1,000 users** covering up to **500 stocks**. It consolidates prior conversations and gives a production-ready design covering:

- system architecture (ingest → RAG → chat),
- model recommendations per task (primary + fallbacks) with numeric quality scores and monthly cost estimates,
- caching strategy and TTLs (including lazy load for SEC filings),
- ingestion, chunking, and embedding rules for filings (10-K, 10-Q, 8-K, 13F), news and social,
- fallback and confidence logic for local/cloud model selection,
- monitoring, security, and deployment considerations,
- sample prompts and API flow snippets.

This design assumes **no local GPU** and uses cloud inference providers (Groq, Together, Gemini, etc.) with aggressive caching and confidence-based fallbacks to minimize cost while preserving high-quality investor-facing outputs.

Core requirements (recap)

- **Users:** 1,000 retail users; each up to 50 stocks; **universe max 500 stocks**.
- **Ingest cadence:** ~100 SEC filings/day (lazy load), news/social batch 3× per day for 500 stocks.
- **Primary tasks:** SEC filing extraction/summarization, news/social sentiment, KPI extraction, portfolio chat.
- **Caching:** filings & summaries TTL = 30 days; news summaries TTL = 24–72 hours; embeddings cached 30 days.
- **Models:** cloud-first. Primary bulk model: Groq GPT-OSS 20B. Chat fallback: Gemini Flash (Gemini Pro for high-value cases).
- **Budget target:** keep recurring cost in the **\$35–\$100 / month** range for inference; total ops may include vector DB, storage, and monitoring.

System architecture (high level)

Components: - Ingestion Service (EDGAR fetcher, News/Social collectors) - Preprocessor & Normalizer (PDF/HTML/XBRL -> Clean text) - Chunker & Tokenizer (1k token target, 150–250 overlap) - Embedding Service (serverless call to chosen embedding model) - Vector Store (FAISS / managed vector DB) - RAG Service (retrieval, prompt assembly) - Inference Layer (Groq/Together for bulk; Gemini for chat) - Cache Layer (Redis/Cloud cache + object storage for raw docs) - API / Chat Frontend (user-facing) - Monitoring & Logging (Prometheus/Grafana, audits)

Flow: 1. New filing request -> check cache -> if miss -> pull from EDGAR -> preprocess -> chunk -> embed -> store -> run bulk summarization -> cache results (30d). 2. News/social (3x/day) -> collect -> dedupe -> preprocess -> chunk -> embed -> run batch sentiment & summarization -> store & cache (24–72h). 3. User chat -> retrieve cached summaries & relevant chunks -> assemble RAG prompt -> call Gemini Flash (or Groq) -> return answer; if low confidence -> escalate to Pro model -> store final answer in cache.

(Include diagrams in your UI or architecture doc as needed.)

SEC filings ingestion & chunking rules

Input formats: PDF, HTML (EDGAR), XBRL.

Steps: 1. Fetch raw filing via EDGAR (or vendor). Store raw file in object storage with metadata (CIK, filing_type, filing_date, accession, hash). 2. Convert to text: pdftotext / Apache Tika / BeautifulSoup for HTML. For XBRL, parse numeric facts to JSON (arelle or sec-xbrl parsing libs). 3. Clean & normalize: remove headers/footers, page markers, normalize whitespace, keep section headings. 4. Tokenize using the target model's tokenizer. Aim for chunk length **~1,000 tokens** with **150–250 token overlap**. Ensure chunk metadata includes `chunk_id`, `filing_id`, `section_heading`, `offset`, `token_count`. 5. Store chunks (text + metadata) in object storage and compute embeddings (embedding model of choice) and insert into vector DB (FAISS/managed).

Chunk meta: filing_id, cik, company_name, filing_type, period_end, chunk_id, section, token_range.

Numeric extraction: run a second pass for tables using XBRL facts and regex numeric extraction. Persist structured KPIs (revenue, net_income, assets, liabilities, guidance statements) as JSON documents in DB and index them for query.

Embeddings and retrieval

- **Embedding model choices:** `bge-base`, `all-mpnet-base-v2`, or `intfloat/e5-large` depending on provider. Use the same embedding model for both filings and news to keep vectors comparable.
- **Index:** FAISS (local or managed) with `IndexHNSWFlat` for performance at scale. Normalize embeddings for cosine similarity.
- **Retieval strategy:** top-k = 8 for summary tasks; top-k = 12 for evidence-heavy answers. Use a re-ranker (cross-encoder) for edge-cases.

Model choices & recommendation table

(See the numeric quality/cost comparison below — primary model followed by 1–2 fallbacks per task.)

Legend: Quality scores are out of 10 (higher = better). Cost columns are estimated monthly inference spend for your workload.

Task	Primary Model	Quality	Monthly Cost (est)	Fallback 1	Fallback 2	Cache TTL
SEC filing extraction	Groq GPT-OSS 20B	9.0	$1.8–2.5	Qwen 14B (Together)	DeepSeek 14B	30 days
Filing long summary	Groq GPT-OSS 20B	8.7	included above	Qwen 20B	DeepSeek 20B	30 days
News summarization	Groq GPT-OSS 20B	9.2	$6–8	Together Qwen 14B	DeepSeek 14B	24–72 hrs
Social sentiment	Groq GPT-OSS 20B	9.2	$4–5	DeepSeek 14B	Qwen 14B	24 hrs
KPI / numeric parsing	Groq / XBRL pipeline	9.0	$2–4	Gemini Flash	Gemini Pro	persistent JSON
Investor chat (RAG)	Gemini Flash	9.0	$15–30	Gemini Pro	OpenAI GPT-4.1	12–24 hrs

> **Total hybrid monthly estimate: \$35–60** (inference only) — storage, vector DB, and API infra additional ~$10–30/mo.

Fallback & confidence logic (production-ready)

Confidence scoring mechanics: - **LLM self-score:** instruct model to output a confidence band (0–1) for each generated factual claim. - **Cross-encoder re-ranker score:** use a small cross-encoder to score retrieval relevance (0–1). - **Numeric match validator:** for numeric claims, run exact/regex lookup in retrieved chunks; if not found, mark as "Not in provided context".

Decision flow (per query): 1. Retrieve top-k evidence & assemble the RAG prompt. 2. Call primary model (Groq for bulk, Gemini Flash for chat) with temperature=0.0. 3. Extract LLM confidence and re-ranker score. 4. If `confidence >= 0.85` and numeric claims match → accept and cache. 5. If `0.6 <= confidence < 0.85` → escalate to higher-quality model (Groq→Qwen20 or Gemini

Flash→Gemini Pro). Re-run and accept if improved. 6. If `confidence < 0.6` or numeric claims mismatch → escalate to premium API (Gemini Pro / GPT-4.1) and mark result for manual review. 7. Log all escalations for feedback loop and model tuning.

Fallback policy examples: - **Filing extraction**: Groq primary; if confidence < 0.6 → Qwen14 (Together); if still low → Gemini Flash. - **Investor chat**: Gemini Flash primary; if low confidence or user flagged → Gemini Pro.

Caching strategy (detailed)

Cache types & TTLs:

- **Filing raw text**: Key = `filing:{cik}:{filing_type}:{period}` → TTL 30 days (or rotate on new filing)
- **Filing chunk embeddings**: Key = `emb_chunk:{filing_id}:{chunk_id}` → TTL 30 days (or persist longer)
- **Filing summaries**: Key = `summary:filing:{cik}:{period}:{model_version}` → TTL 30 days
- **KPI JSON (parsed)**: Key = `kpi:filing:{cik}:{period}` → persistent until replaced
- **News digest per ticker**: Key = `news:{ticker}:{run_timestamp}` → TTL 24–72 hours
- **Social digest per ticker**: Key = `social:{ticker}:{run_timestamp}` → TTL 6–24 hours
- **User portfolio snapshot**: Key = `portfolio:{user_id}:{date}` → TTL 24 hours
- **RAG answer cache**: Key = `rag_answer:{query_hash}:{model_version}` → TTL 12–24 hours

Cache invalidation rules: - If a new filing appears for a CIK, invalidate `summary:filing:*` and `rag_answer:*` keys referencing that filing. - If embedding/index rebuild happens, mark embeddings expired and recompute lazily. - Use model_version in cache keys to ensure changes to model prompts/ models invalidates old cached outputs.

Prompt templates (samples)

System prompt (filing summarizer):

```
You are FinSumm-Assistant, an expert financial document summarizer. Use only the
provided excerpts. For each factual statement, include the source chunk id. If a
number or claim is not present in the retrieved context, say "not in provided
context." Output: short headline, 5 key bullets (with chunk ids), material
risks, follow-ups, suggested action.
```

User prompt (investor chat): `User portfolio: [ticker list]. Use cached summaries and latest news digests. Answer: 1-sentence thesis, 3 supporting facts (cite chunk ids), risk note, suggested action. If uncertain, ask the user to approve a deeper analysis (escalate to premium).`

API flow snippets (pseudo)

1. `GET /filing/{cik}/{type}/{period}` -> check cache -> if miss -> fetch EDGAR -> preprocess -> chunk -> embed -> store -> summarize -> return.
2. `POST /batch/news` -> enqueue -> dedupe -> chunk -> embed -> batch infer (Groq) -> store digests.
3. `POST /chat/query` -> retrieve user snapshot & relevant chunks -> assemble RAG -> call Gemini Flash -> if low confidence -> escalate -> return.

Monitoring, metrics & feedback loops

Track these KPIs: - Cache hit rate (target > 85%) - Average inference cost per summary - Fallback rate (target < 10%) - Hallucination flags / numeric mismatch rate - Latency P50/P95 - User satisfaction / upvote rate

Feedback loop: - Collect flagged outputs for manual curation. - Periodically fine-tune re-ranker or small classifiers to reduce fallbacks.

Security, compliance & governance

- Encrypt S3 buckets at rest, use signed URLs.
- PII handling: strip user PII from logs; separate telemetry and content storage.
- Retention: filings and parsed KPIs kept per TTL policy; keep an audit log of LLM outputs for 90 days.
- Rate limit API keys, use service accounts and least-privilege IAM roles.

Deployment & cost ops

- Use serverless functions / Kubernetes for workers.
- Use managed vector DB (Pinecone/Chroma/Weaviate) for reliability at scale if budget allows; FAISS on ephemeral nodes is OK for cost saving.
- Expected inference + vector costs: **$35–60/month** for the hybrid setup (Groq + Gemini Flash) + storage and DB ($10–30/month).

Next steps / deliverables I can produce

- Full **FastAPI microservices** skeleton for ingestion, RAG, and chat endpoints (complete with Redis caching).
- **Dockerfile + deployment scripts** for cloud provider of choice.
- **Cost calculator spreadsheet** to tune model splits (Groq vs Together vs Gemini).
- **Benchmark harness** for your sample filings & news to measure real fallback rates.

Document prepared by: ChatGPT (assistant).