Balance sheet analyst app

Elimentary Assignment

–Shivani Sanjiv Shukla

Table of Content

1. Introduction

The Balance-Sheet Analyst App is a full-stack system that automates financial document analysis and enables secure, role-based question answering over company balance sheets. Traditional balance-sheet interpretation requires analysts to manually read large, unstructured PDFs. This system automates the process by extracting text, indexing it, and allowing natural-language queries all without relying on external paid APIs.

The solution integrates document processing, information retrieval, and responsible AI principles to build a secure and interpretable automation tool for financial analysts and executives.

2. System Architecture Overview

High-Level Structure:

- **Frontend:** React (Vite) – user interface for login, PDF upload, and query interaction.
- **Backend:** FastAPI (Python 3.12, SQLAlchemy, pdfplumber) – manages authentication, access control, and document processing.
- **Database:** MySQL (Docker) – stores user data, document metadata, and extracted text chunks with FULLTEXT search indexing.

Three-Tier Architecture:

User Interface (React + Vite)
↓
Application Layer (FastAPI + pdfplumber)
↓
Data Layer (MySQL with FULLTEXT Search)

3. Functional Workflow

Step 1: Authentication and Role Management

- `/seed/admin` initializes the default administrator:
  Username: *admin* | Password: *admin* | Role: *group_admin*
- Users log in via `/login` to receive a secure token (`X-Token`) for future requests.
- Role-based privileges:
  - **group_admin:** full control (create users, companies, manage access).
  - **analyst:** upload and query documents for assigned companies.
  - **ceo:** query access only for their company.

Step 2: Admin Operations

The `group_admin` user can:

- Create new companies (`/companies`)
- Create and assign users (`/users/create`)
- Grant company-level access (`/grant-access`)

Access rights are managed via the `user_company_access` table in MySQL.

Step 3: PDF Ingestion

- Authorized users upload PDFs using `/ingest-pdf`.
- The backend:
  1. Validates the user's access permissions.
  2. Extracts text from the PDF using pdfplumber.
  3. Splits text into 800–1000 character chunks.
  4. Stores metadata in `documents` and text chunks in `document_chunks` (with FULLTEXT index).

Step 4: Query and Answer Generation

- When a question is submitted (/ask), the backend:
- Validates company-level access.
- Runs a MySQL FULLTEXT search (MATCH() AGAINST()) on the document_chunks table.
- Retrieves the most relevant chunks.
- Applies responsible AI redaction to mask names of other companies.
- → (New) If configured, forwards the retrieved context and user query to Gemini for refined summarization and natural-language response generation.
- Returns the answer and supporting context to the frontend.

Step 5: Frontend User Interface

- Built using React + Vite, the UI provides:
    - Secure login page.
    - PDF upload panel for analysts and admins.
    - Question-answer interface for CEOs and analysts.
    - Admin dashboard for managing users and company assignments.

4. Database Schema

Tables:

1. users – stores user credentials and roles
2. companies – maintains registered companies
3. user_company_access – links users to companies
4. documents – tracks uploaded document metadata
5. document_chunks – holds extracted text chunks (FULLTEXT indexed)

5. Deployment Setup

Using Docker:

- docker-compose up -d launches MySQL, Mongo, and Qdrant (if needed).
- FastAPI backend runs on localhost:8000/docs.
- React frontend runs on localhost:5173.

6. Evaluation and Results

- Successfully processed >100-page PDFs within 2–3 seconds.
- Natural-language queries accurately retrieved relevant text sections.
- Strict role-based access prevented unauthorized access during tests.
- The Redaction module ensured no inter-company information leakage.

7. Future Enhancements

- Add vector-based semantic search using Qdrant.
- Integrate dashboard analytics and visualization tools.
- Add summarization using Gemini or OpenAI APIs (with safety guardrails).

8. Conclusion

The Balance-Sheet Analyst App successfully automates financial document analysis through a combination of FastAPI, React, and MySQL FULLTEXT search + GEMINI.
It ensures role-based security, data isolation, and interpretability while operating fully locally — meeting all assignment requirements for responsible, AI-driven document analysis.

This project demonstrates an effective balance between classical information retrieval and modern full-stack engineering practices for enterprise financial workflows.