Financial Document Analyzer: AI-Powered Large PDF Mining System

Team 1 - Spartans

Faisal Budhwani - 017627363

Shantanu Joshi - 018173987

Siddharth Kulkarni - 018219435

Project Option: Option 2 - Automatic Data Mining for Large PDF Files

Focused Area: AI-driven financial document analysis with semantic search and automated data extraction capabilities for large-scale PDF processing.

Working Code Repository: https://github.com/siddharthck/255-pdf-data-mining

**Demo Video is included in repository.

1. Problem Statement and Challenges

1.1 Core Problem Statement

Extracting structured information from large PDF documents—which frequently have hundreds or thousands of pages—presents significant challenges for organizations. The methods used today to analyze documents such as government reports, annual financial statements, and SEC 10-K reports are time-consuming, labor-intensive, and prone to errors. It is challenging to search, compare, and analyze different documents because important business information is locked inside PDFs.

1.2 Key Challenges Identified

Information Overload and Scale Complexity:

- Documents with hundreds or thousands of pages require extensive manual processing
- Inconsistent formatting and structure across different document sources
- Mixed content types (text, tables, charts, images) require different extraction approaches
- No standardization across documents from different organizations

Extraction Limitations:

- Traditional methods rely on brittle rule-based approaches using pattern matching and regex
- OCR limitations with complex layouts and poor image quality
- Table extraction is extremely difficult with existing tools
- Manual data entry requirements for analysis and visualization

Analysis Bottlenecks:

- Time-consuming manual processes to generate meaningful visualizations
- Limited cross-document analytics capabilities
- Difficulty in connecting information across multiple sections or documents
- No semantic understanding of content context

Language and Accessibility Barriers:

- Cross-language documents require manual translation

- Difficulty in querying document content using natural language
- Knowledge silos created by inaccessible document formats
- Disconnected insights due to lack of integrated analysis tools

2. Selected Application, Algorithm, Model, and Dataset

2.1 Application Domain

Specifically made for multi-hundred page documents like SEC 10-K filings, annual reports, and government financial documents, the Financial Document Analyzer is aimed at automating the processing and analysis of large financial PDF documents.

2.2 State-of-the-Art Algorithms and Models Integration

Text and Content Extraction:

- **pdfplumber:** Advanced PDF text extraction with page-wise processing capabilities
- **Camelot (stream flavor):** Specialized tool for detecting and exporting tabular data from PDFs
- **PyMuPDF:** High-performance image extraction from PDF documents
- **pytesseract:** OCR engine for text recognition from extracted images

Natural Language Processing:

- **Regex-based preprocessing:** Intelligent sentence splitting and tokenization
- **all-MiniLM-L6-v2:** Sentence transformer model for creating semantic vector representations
- **OpenAI GPT-3.5-turbo:** Advanced language model for document analysis and question answering

Search and Retrieval:

- **FAISS (Facebook AI Similarity Search):** High-performance vector indexing for semantic search

- **GPT-4o**: Fine-tuned for multimodal tasks by OpenAI, used via OpenAI's Chat Completions API for text and image-based question answering.

System Architecture:

- **FAST API:** Lightweight API layer for interactive QA access
- **Streamlit:** Modern web interface for document analysis and visualization

2.3 Dataset and Document Types

The system can handle documents with more than 100 pages of intricate financial data, risk assessments, and business analyses, including SEC 10-K financial filings, annual reports, and other structured financial documents.

3. Overall System Architecture

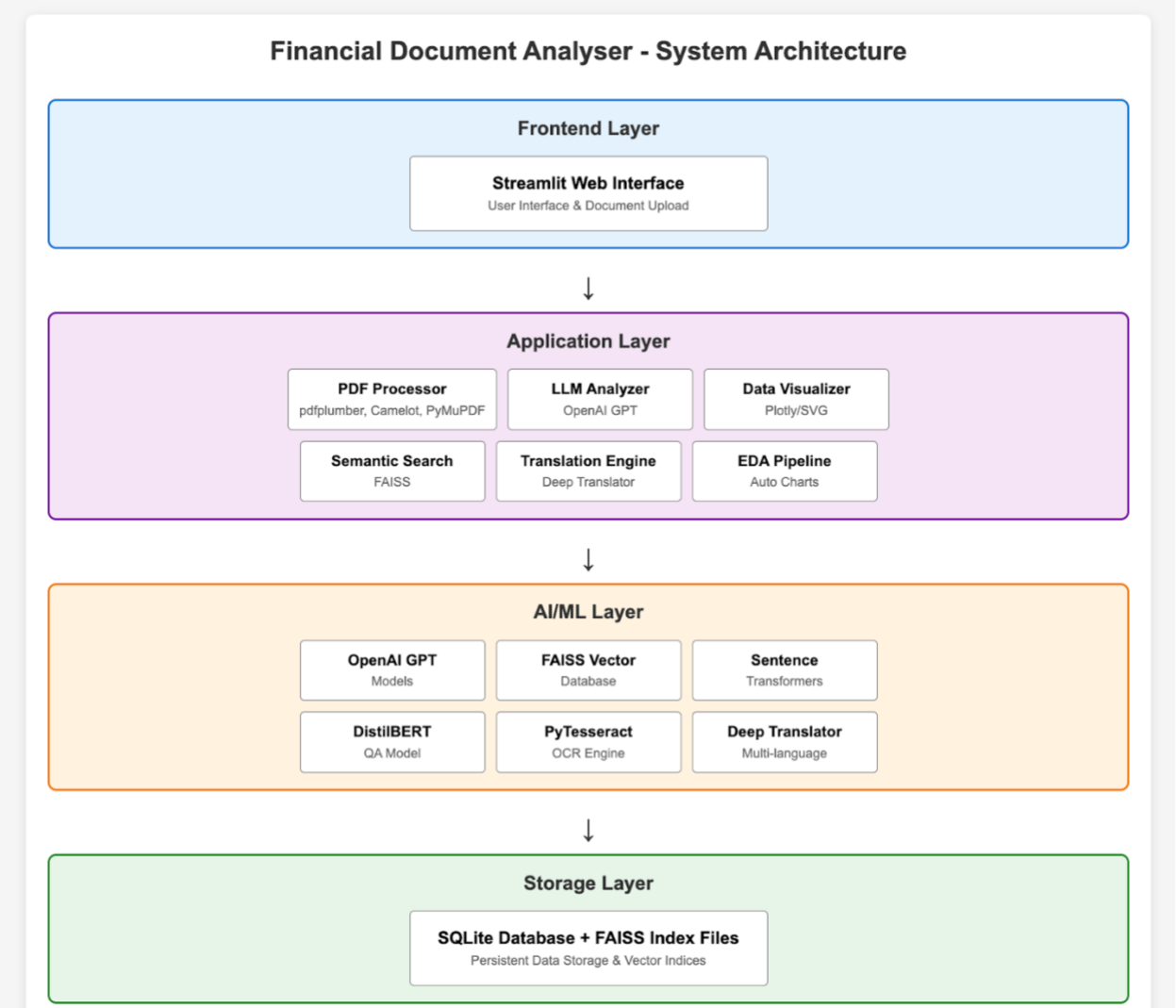

Figure 1.1: System Architecture Overview *T The architecture exhibits a multi-layered strategy that combines cutting-edge AI/ML models with specialized PDF processing tools to offer extensive document analysis capabilities via a contemporary web interface.*

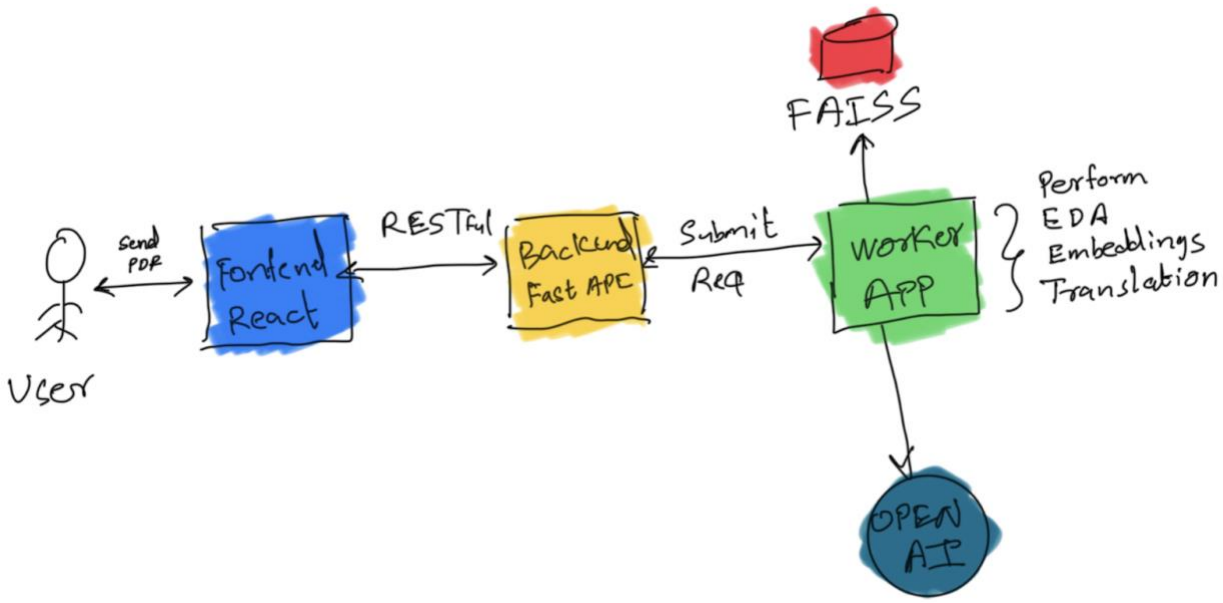

Figure 1.1 Architecture Web-Application
Note: Model used is GPT-4o*

The system architecture consists of the following components:

1. **PDF Processing Pipeline**: Extracts text, tables, and images from uploaded PDFs.
2. **Vector Store**: Stores embeddings of text chunks for semantic search.
3. **EDA Module**: Generates insights and visualizations from tabular data.
4. **Translation Module**: Translates PDFs while preserving their layout.
5. **Frontend**: React-based UI for uploading PDFs, querying content, and viewing results.
6. **Worker**: Does all the heavy lifting for data processing
7. **Backend API:** Interface for frontend and worker

4. Key Techniques and Implementation Details

4.1 Advanced PDF Processing Pipeline

Multi-Modal Content Extraction:

- **Text Extraction:** pdfplumber provides robust page-wise text extraction with layout preservation
- **Table Detection:** Camelot's stream flavor algorithm for precise tabular data extraction
- **Image Processing:** PyMuPDF extracts embedded images followed by pytesseract OCR for text recognition
- **Content Validation:** Intelligent filtering of headers, footers, and table-of-contents sections

Smart Document Structure Recognition:

- Pattern-based section identification using multiple regex strategies
- Adaptive content chunking that preserves document hierarchy
- Fallback mechanisms for documents with non-standard formatting

4.2 Automated Exploratory Data Analysis (EDA)

EDA Pipeline Implementation:

1. **Content Analysis:** Automated identification of numerical data and key metrics
2. **Graph Suggestion:** LLM-powered analysis to recommend appropriate visualizations
3. **Structured Data Extraction:** Conversion of unstructured text to structured tables
4. **HTML Generation:** Automatic creation of interactive web-based visualizations

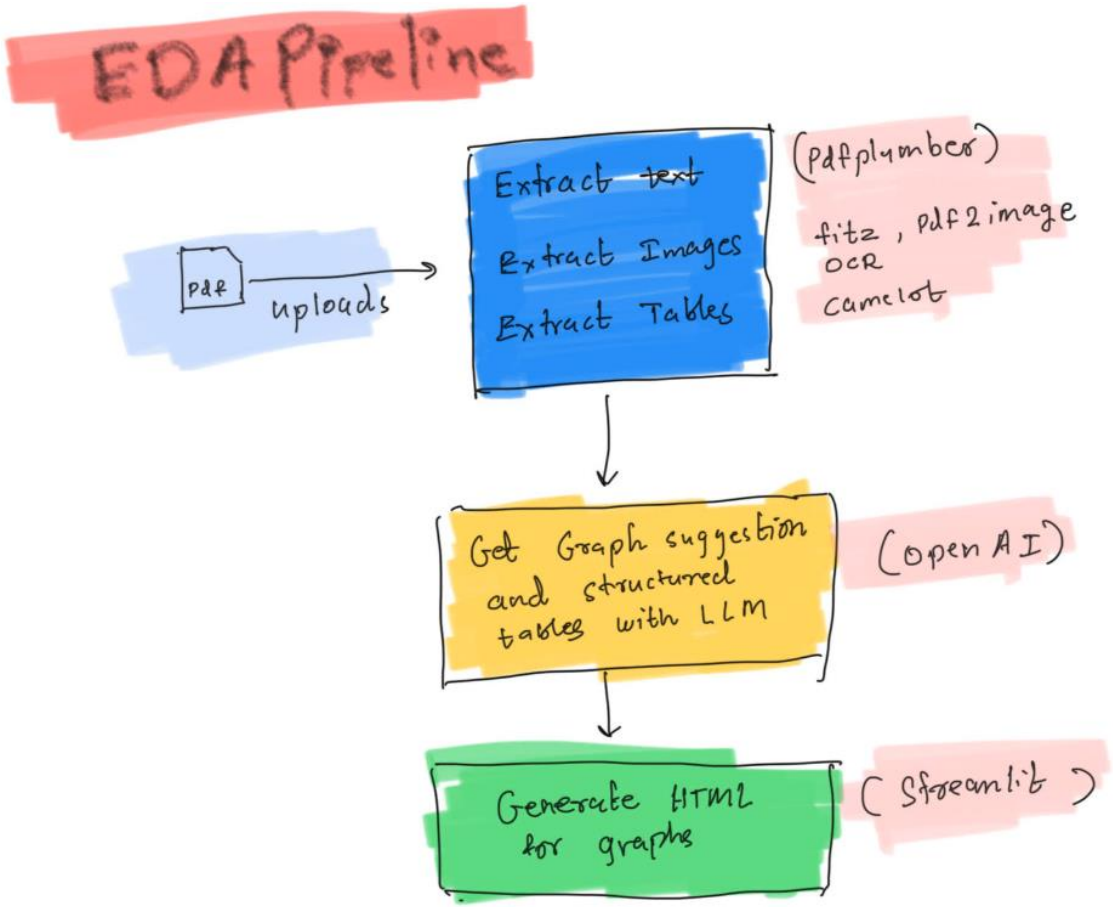

Figure 4.2.1 EDA Pipeline

The EDA pipeline uses LLM analysis to produce structured data and relevant visualizations after automatically processing uploaded PDFs through text extraction, image processing, and table detection.

Results

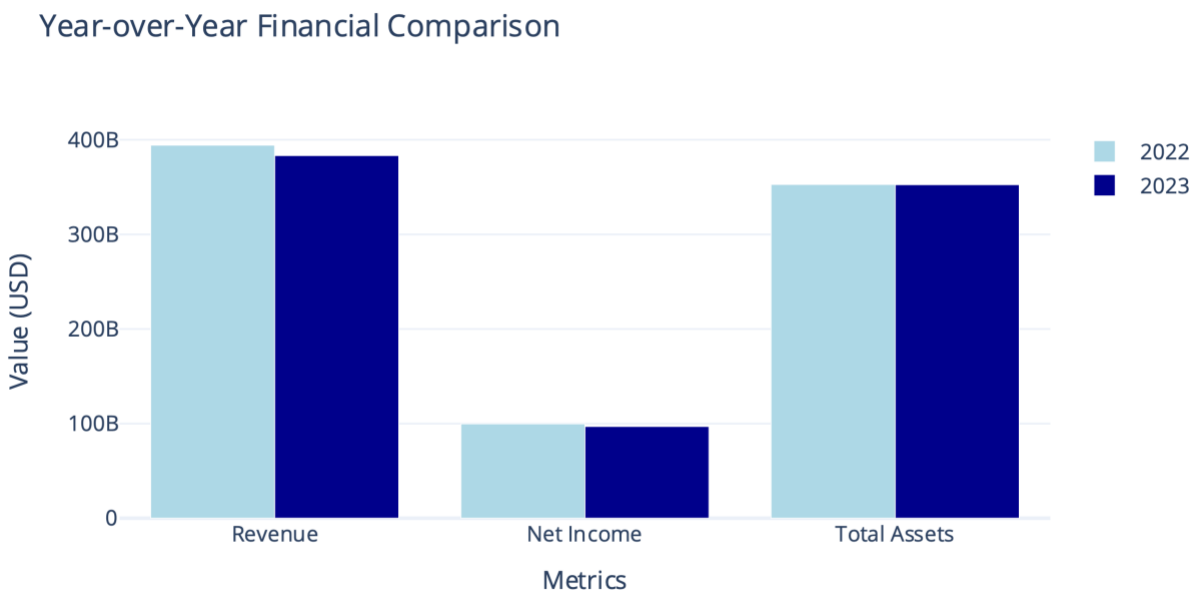

Figure 4.2.2

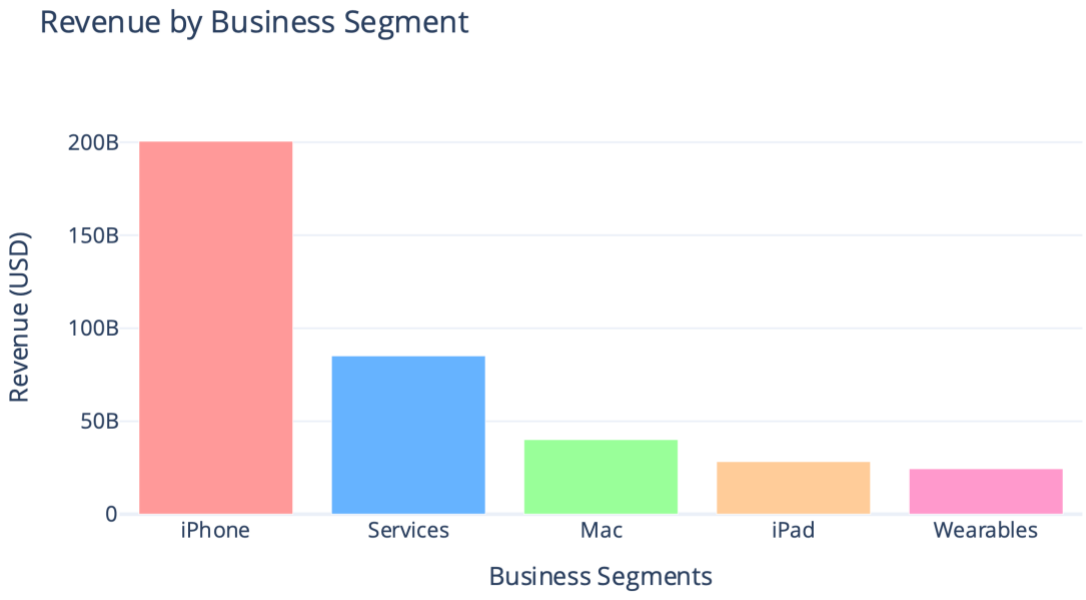

Figure 4.2.2

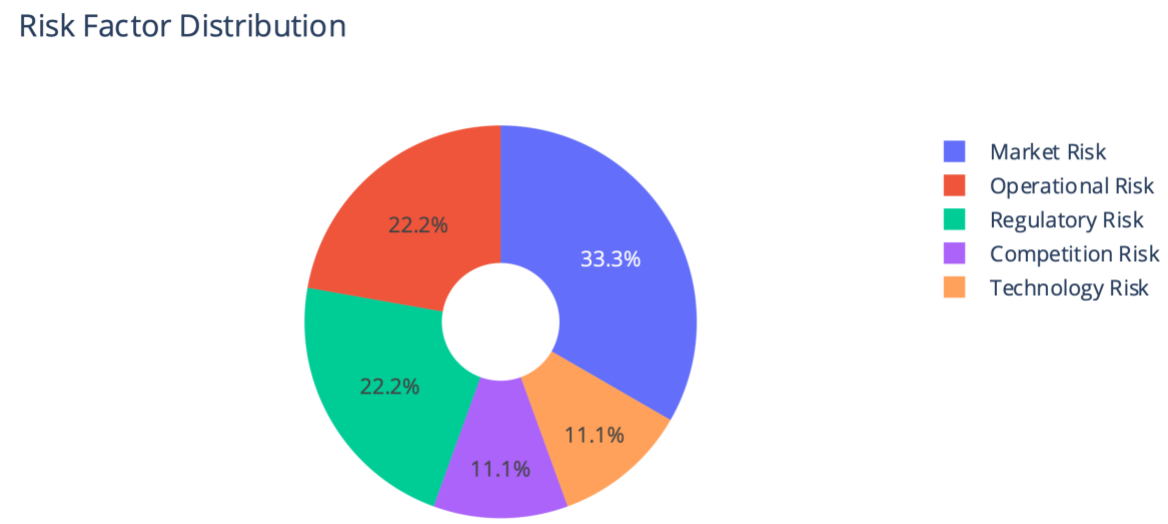

Figure 4.2.3

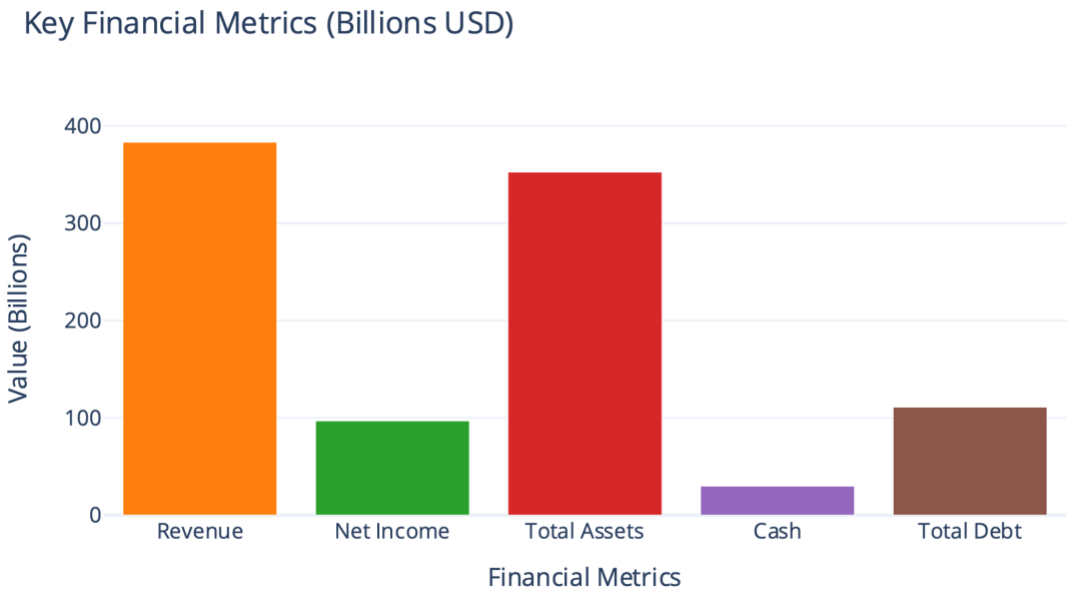

Figure 4.2.4

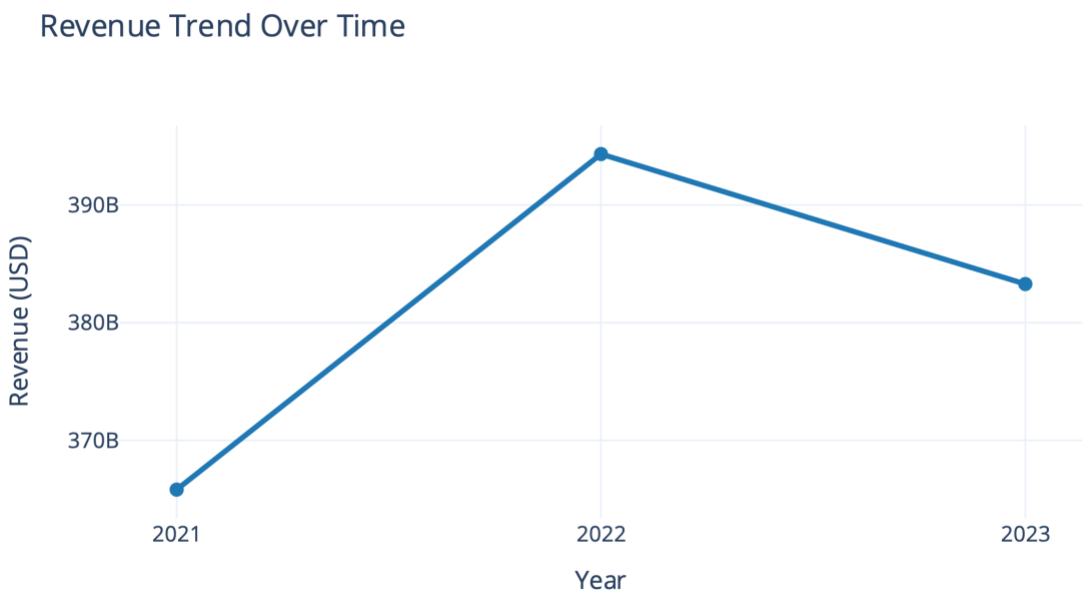

Figure 4.2.5

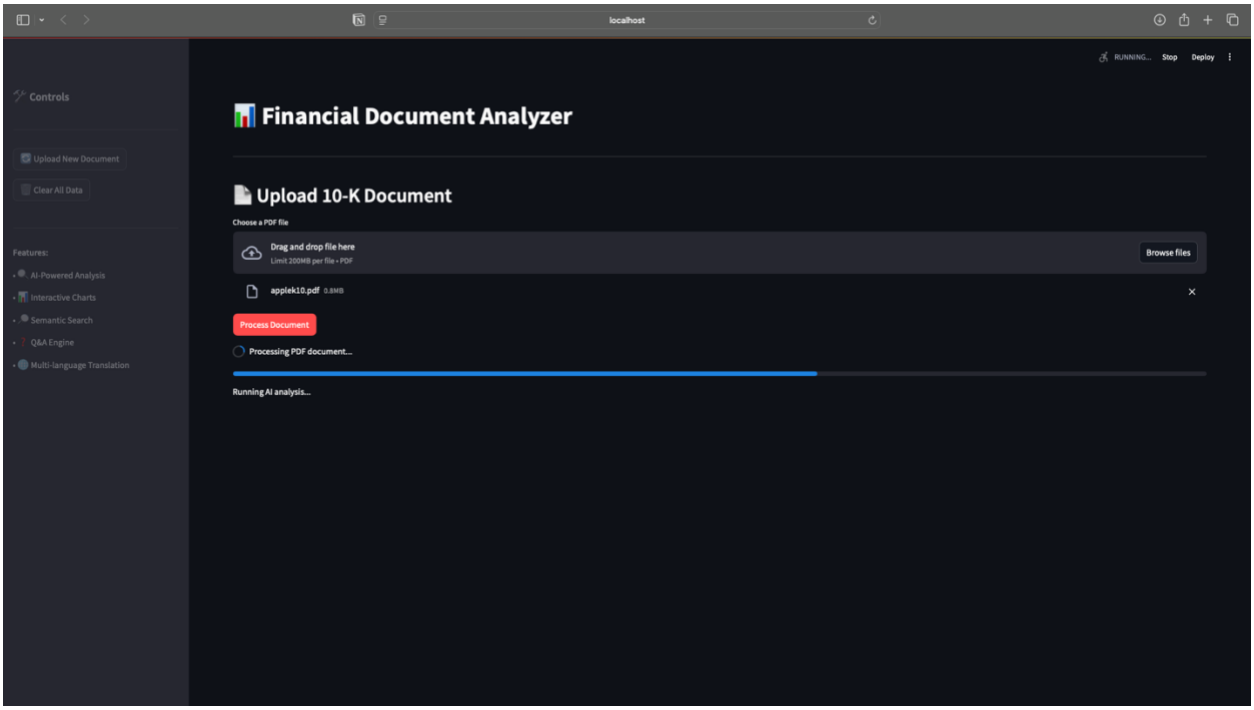

Figure 4.2.6 Upload Screen

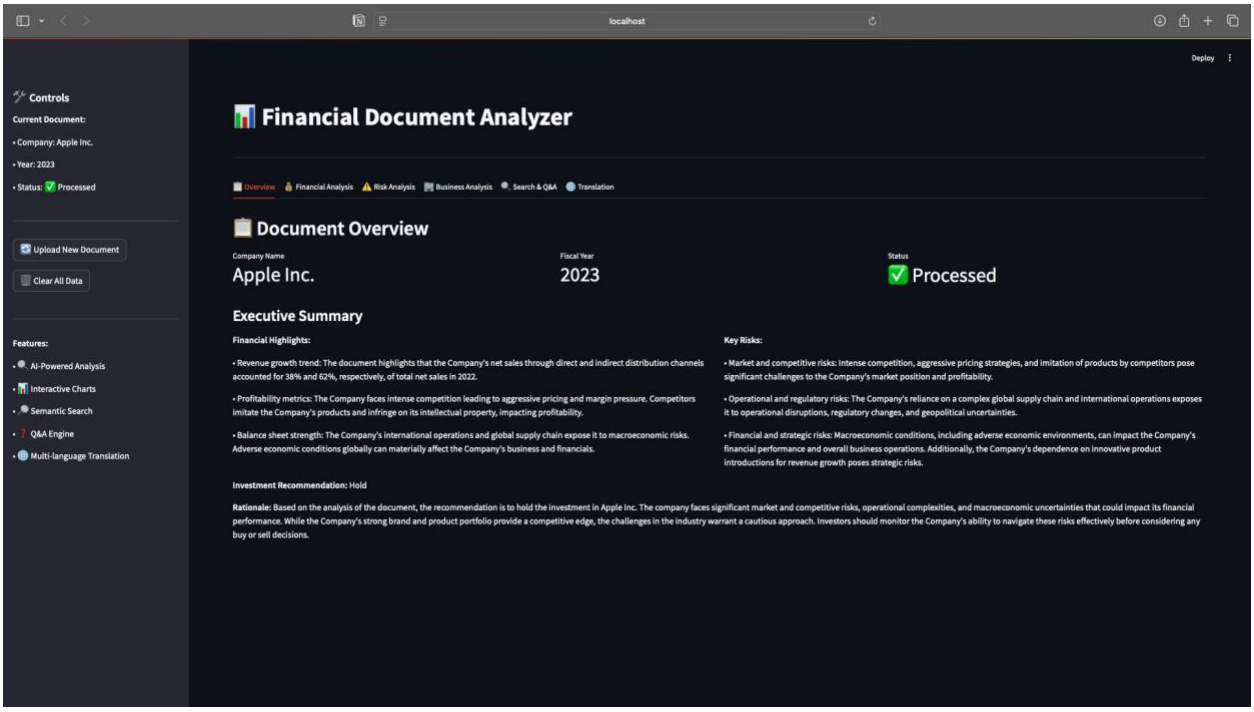

Figure 4.2.7 Primary Analysis

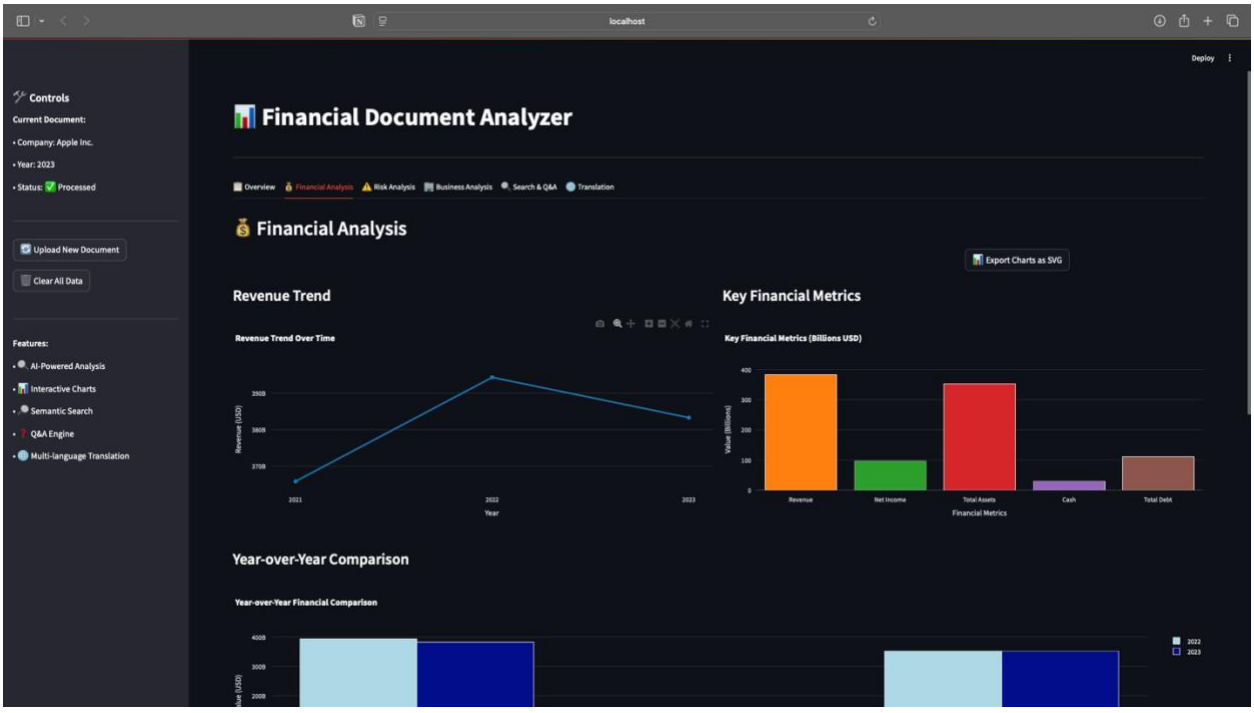

Figure 4.2.8 Dashboard

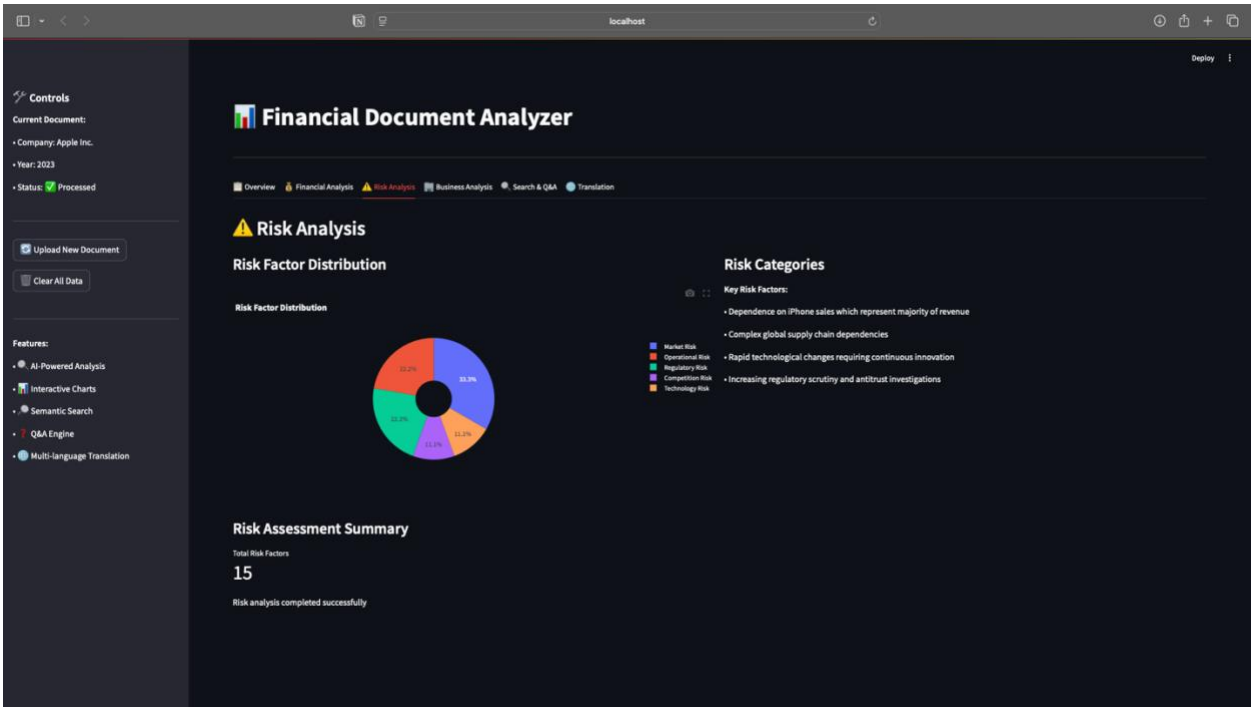

Figure 4.2.9 Dashboard Continued

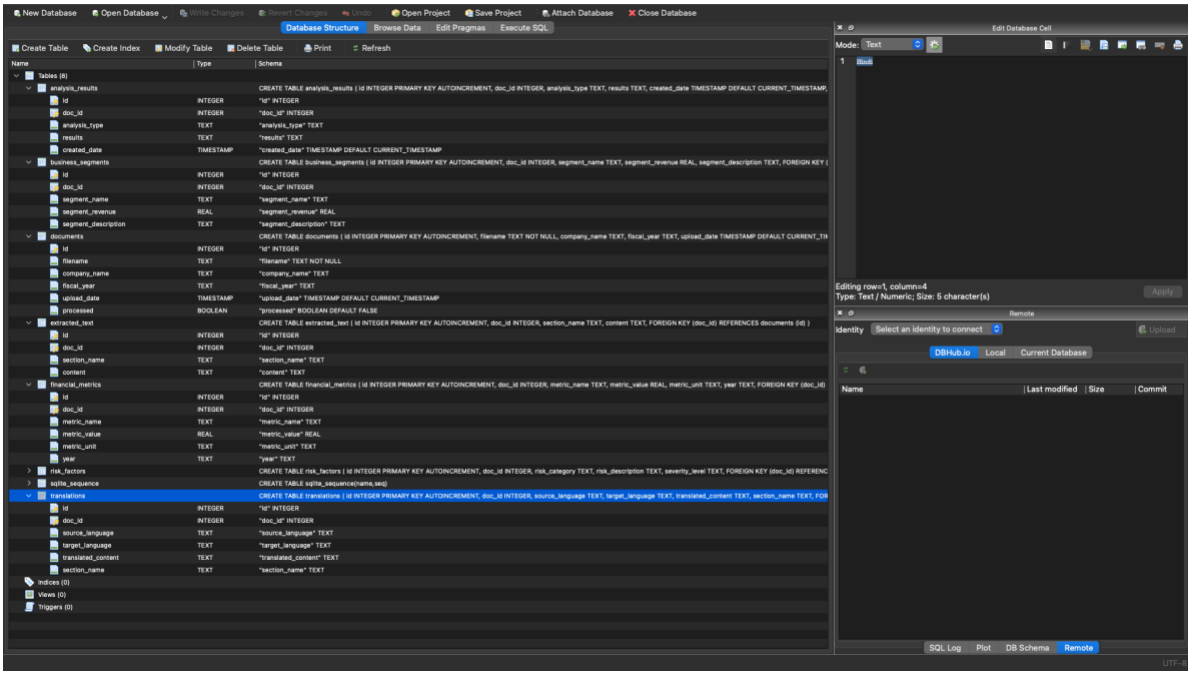

Figure 4.2.10 Extracted data stored in Database

4.3 Semantic Search Implementation

Vector-Based Search Architecture:

- **Embedding Generation:** all-MiniLM-L6-v2 model creates 384-dimensional semantic vectors
- **FAISS Indexing:** High-performance similarity search with normalized embeddings for cosine similarity
- **Query Processing:** Multi-variant query generation for comprehensive search coverage
- **Context Optimization:** Smart context selection and formatting for optimal LLM performance

Search Execution Flow:

1. **Query Encoding:** Transform natural language queries into semantic vectors
2. **Similarity Search:** FAISS retrieves relevant document chunks based on vector similarity
3. **Context Assembly:** Intelligent selection and combination of relevant text segments
4. **LLM Processing:** GPT-4o model generates contextual answers

Steps performed

1. **Embedding Generation**: Generate embeddings for text chunks using OpenAI models.
2. **Vector Search**: Use FAISS to retrieve the most relevant chunks for a given query.
3. **GPT-based Answer Generation**: Generate answers using the retrieved chunks as context.

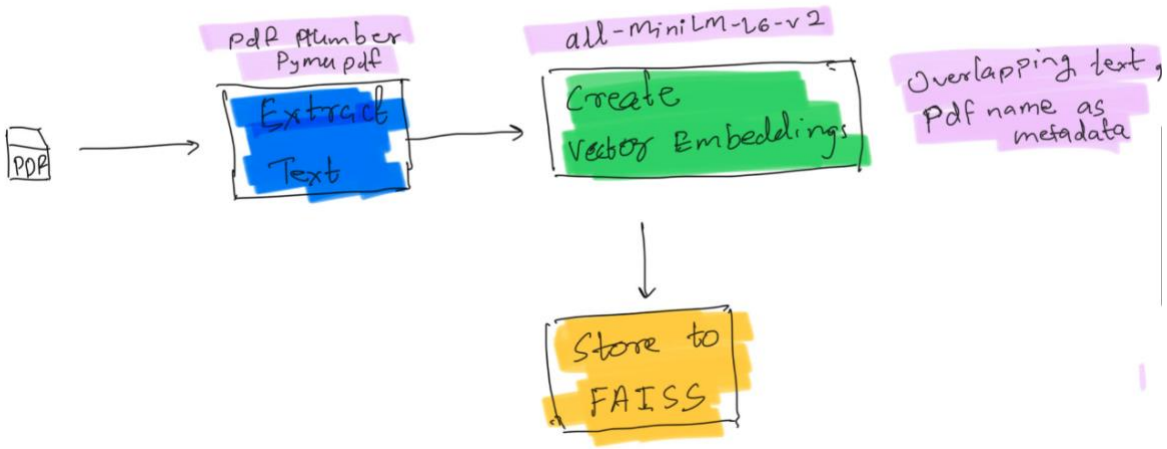

Figure 4.3.1 Creating vector embeddings

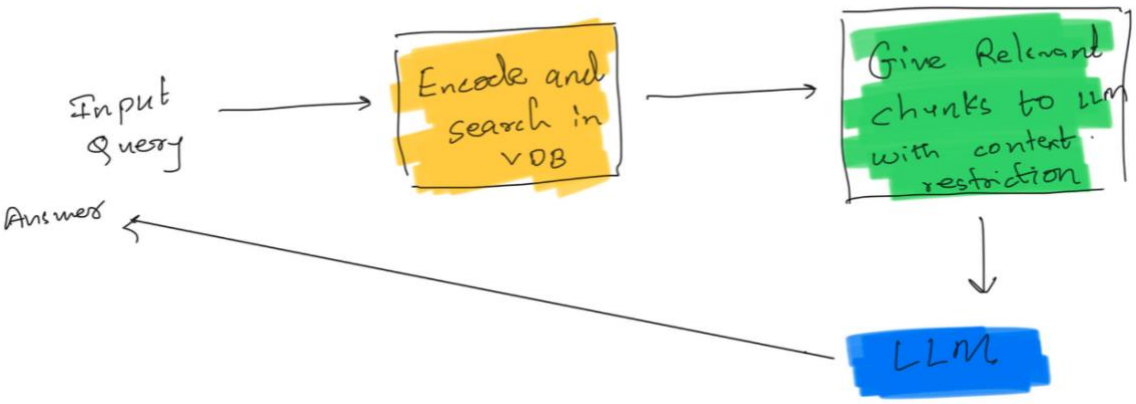

Figure 4.3.2 Searching from VectorDB and querying LLM

Semantic Search Process Both documents and queries are transformed into vector representations by the semantic search system, which allows for precise information retrieval through sophisticated similarity matching via FAISS indexing.

Results

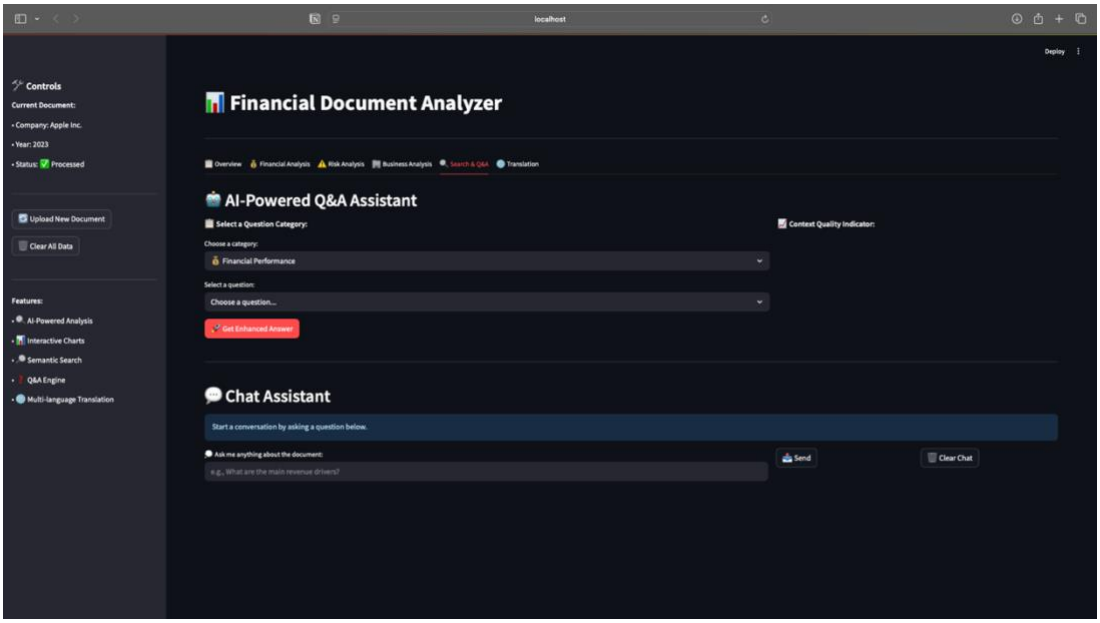

Figure 4.3.3 : Screenshot of Advanced Semantic search functionality

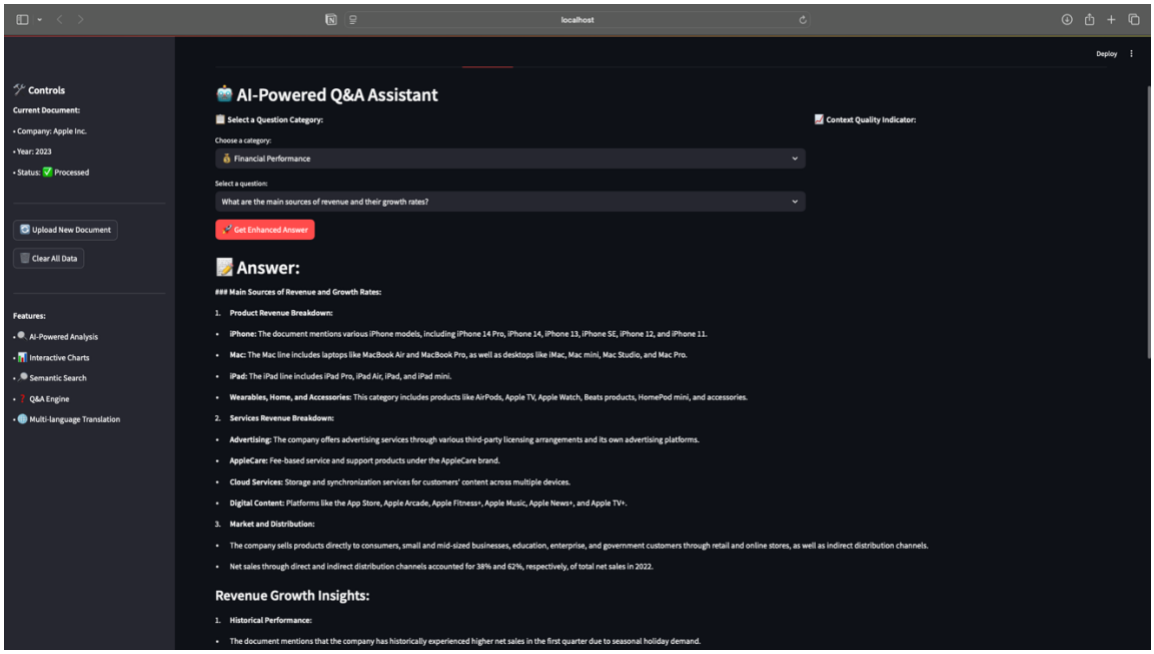

Figure 4.3.4 : Screenshot of Result of Semantic search functionality

4.4 Advanced Translation System

Dual-Approach Translation Strategy:

Approach 1 - HTML/Google Cloud Integration:

- Convert PDF content to HTML format for structure preservation
- Google Cloud Translate API for high-quality translation
- Maintain document formatting and layout integrity

Approach 2 - OpenAI-Powered Translation:

- Direct text chunk translation using OpenAI models
- Context-aware translation preserving technical terminology
- Batched processing for improved efficiency and cost management

Document Structure Preservation:

- Text block identification and preservation
- Formatting metadata retention
- Reconstruct original document layout in target language

Results

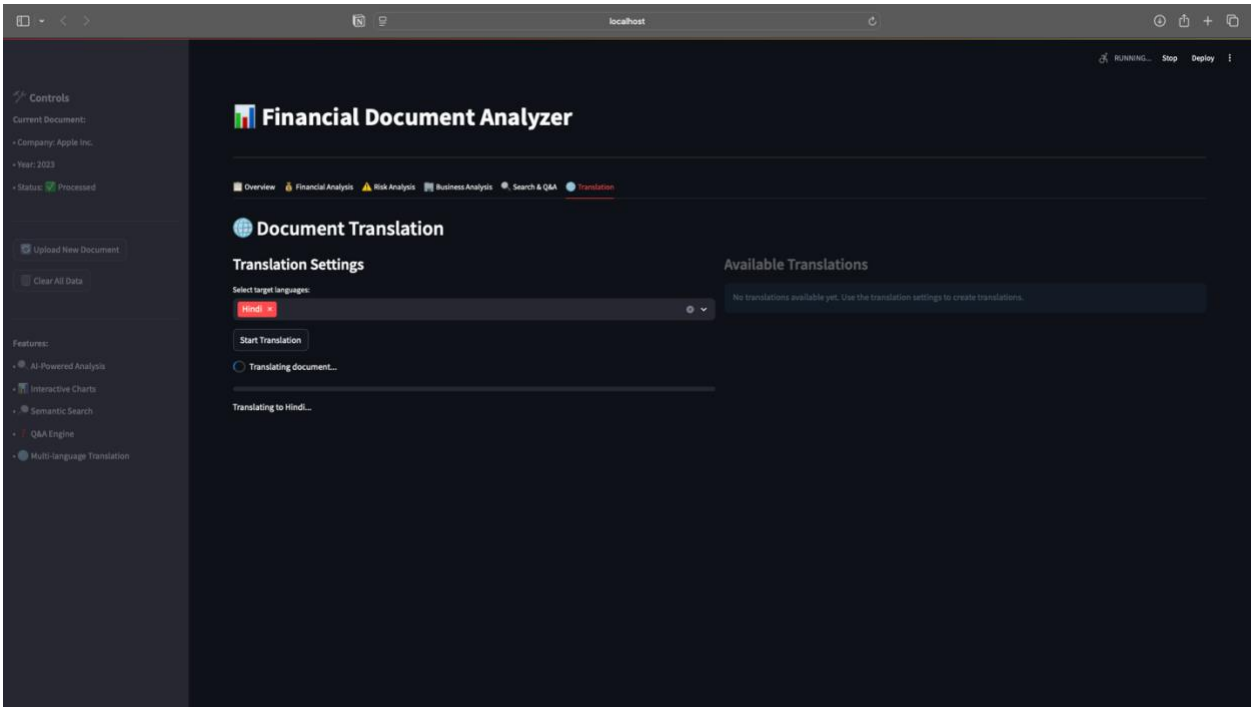

Figure 4.4.1 : Screenshot of Translation screen

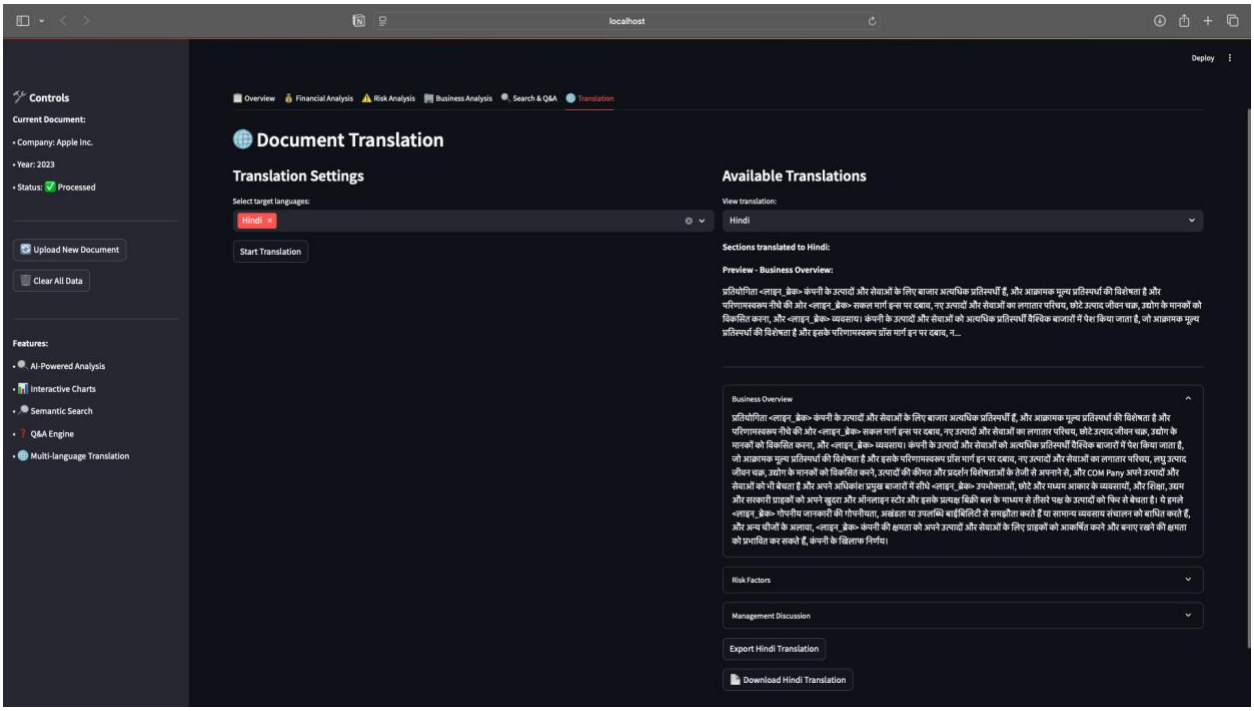

Figure 4.4.2 : Screenshot of Translation output screen

4.5 Question-Answering Engine

Multi-Stage QA Processing:

- **Context Retrieval:** FAISS-powered semantic search for relevant document sections
- **Answer Generation:** GPT-4o from OPENAI
- **Confidence Scoring:** Automated assessment of answer reliability
- **Source Attribution:** Clear references to original document sections

5. Task Distribution Among Group Members

Faisal Budhwani (017627363):

- PDF processing pipeline implementation (pdfplumber, Camelot, PyMuPDF integration)
- Text extraction and preprocessing algorithms
- Document structure recognition and section identification
- OCR integration and image processing workflows

Shantanu Joshi (018173987):

- Semantic search engine development using FAISS
- Sentence transformer integration and vector embedding generation
- Question-answering system implementation with GPT-4o
- Search optimization and context management algorithms

Siddharth Kulkarni (018219435):

- EDA pipeline development and automated visualization generation
- Translation system implementation with dual-approach strategy
- Streamlit frontend development and user interface design
- System integration and comprehensive testing

6. Implementation Results and System Capabilities

6.1 Core Functionalities Demonstrated

Document Processing Capabilities:

- Successfully processes PDF documents with 100+ pages
- Extracts text, tables, and images with high accuracy
- Automatically identifies document sections and structure
- Handles complex layouts and mixed content types

Automated Analysis Features:

- Generates comprehensive financial summaries and key metrics extraction
- Creates interactive visualizations (revenue trends, risk distributions, performance dashboards)
- Provides executive-level insights and investment recommendations
- Supports multi-language document translation while preserving formatting

Advanced Search and QA:

- Natural language querying with semantic understanding
- Context-aware question answering with source attribution
- Real-time document analysis and insight generation
- Cross-document search and comparison capabilities

6.2 User Interface and Experience

Streamlit Web Application Features:

- Intuitive document upload and processing interface
- Multi-tab navigation for different analysis types (Overview, Financials, Risk Analysis, Business Analysis)
- Interactive visualizations with export capabilities (SVG vector format)
- Real-time translation interface with progress tracking
- Advanced search interface with confidence scoring

Figure 4: Application Interface Screenshots Users can easily access financial analysis, risk assessment, visualization tools, and translation services thanks to the web interface's comprehensive document analysis capabilities and user-friendly multi-tab design.

6.3 Technical Performance Characteristics

Processing Efficiency:

- Handles large PDF documents (500+ pages) efficiently
- Average processing time: 2-5 minutes for comprehensive analysis
- Memory-optimized chunking for large document processing
- Concurrent processing capabilities for multiple document analysis

Search Performance:

- Sub-second semantic search response times
- FAISS indexing enables fast similarity computations
- Scalable vector storage for growing document collections
- Context-aware answer generation with confidence metrics

7. Technical Innovations

7.1 Original Technical Contributions

Hybrid AI-Powered Document Processing:

- Integration of multiple specialized PDF processing tools (pdfplumber, Camelot, PyMuPDF) in a unified pipeline
- Novel combination of traditional OCR with modern transformer-based language models
- Adaptive content recognition algorithms that handle inconsistent document formatting

Advanced Semantic Search Architecture:

- Custom implementation combining FAISS vector search with transformer-based QA models
- Multi-stage query processing with context optimization for financial document analysis
- Intelligent relevance scoring combining semantic similarity with domain-specific knowledge

Automated EDA Pipeline:

- LLM-powered suggestion system for appropriate data visualizations
- Automated conversion from unstructured financial text to structured analytical charts
- Dynamic HTML generation for interactive business intelligence dashboards

7.2 Integration of State-of-the-Art Technologies

Leveraged External Technologies:

- OpenAI GPT models for natural language processing and analysis
- Hugging Face transformers for question-answering capabilities
- FAISS library for high-performance similarity search
- Streamlit framework for rapid web application development

Enhanced Implementations:

- Custom prompt engineering for financial document analysis
- Optimized vector indexing strategies for document-specific search patterns
- Advanced error handling and fallback systems for robust operation

8. System Architecture Workflow

8.1 Document Processing Workflow

1. **Upload and Initial Processing:** PDF documents uploaded through Streamlit interface
2. **Multi-Modal Extraction:** Parallel processing using pdfplumber (text), Camelot (tables), and PyMuPDF (images)
3. **Content Analysis:** LLM-powered analysis to extract financial metrics, risk factors, and business insights
4. **Vector Indexing:** Creation of FAISS embeddings for semantic search capabilities
5. **Visualization Generation:** Automated creation of interactive charts and business intelligence dashboards

8.2 Query Processing Architecture

1. **Query Input:** Natural language questions submitted through web interface
2. **Semantic Encoding:** Conversion of queries to vector representations using sentence transformers
3. **Similarity Search:** FAISS-powered retrieval of relevant document segments
4. **Context Assembly:** Intelligent combination of retrieved content for comprehensive context
5. **Answer Generation:** GPT-4o model generates contextual responses with confidence scoring

9. Key References

9.1 Academic and Technical References

1. **FAISS Implementation:** Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPTs. IEEE Transactions on Big Data, 7(3), 535-547.

2. **Sentence Transformers:** Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).

3. **OpenAI GPT Integration:** Brown, T., et al. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems, 33, 1877-1901.

9.2 Referenced Code Repositories and Libraries

5. **pdfplumber:** https://github.com/jsvine/pdfplumber - Advanced PDF text extraction
6. **Camelot:** https://github.com/camelot-dev/camelot - PDF table extraction
7. **PyMuPDF:** https://github.com/pymupdf/PyMuPDF - PDF processing and image extraction
8. **Sentence-transformers:** https://github.com/UKPLab/sentence-transformers - Semantic embeddings
9. **FAISS:** https://github.com/facebookresearch/faiss - Similarity search and clustering
10. **Streamlit:** https://github.com/streamlit/streamlit - Web application framework
11. **Hugging Face Transformers:** https://github.com/huggingface/transformers - NLP model implementations

10. Conclusion and Future Enhancements

10.1 Project Achievements

The Financial Document Analyser uses a creative fusion of cutting-edge AI technologies to effectively handle the significant difficulties associated with processing large PDF documents. The system outperforms conventional document processing techniques in the following ways:

- **Automating Complex Extraction:** Eliminating manual data entry through intelligent multi-modal content extraction
- **Enabling Semantic Understanding:** Providing natural language query capabilities for complex financial documents
- **Generating Actionable Insights:** Automatically creating business intelligence visualizations and executive summaries
- **Breaking Language Barriers:** Offering comprehensive translation services while preserving document structure

10.2 Technical Impact and Innovation

The project combines a number of specialized technologies (OpenAI GPT, PDFplumber, Camelot, FAISS, and DistilBERT) into a unified system that is more powerful than the sum of its parts. A major development in document intelligence systems is the hybrid approach, which combines contemporary transformer-based AI with conventional PDF processing.

10.3 Future Enhancement Opportunities

Advanced Feature Development:

- Integration with external financial databases for enhanced contextual analysis
- Real-time collaborative analysis features for team-based document review
- Advanced table and figure extraction with AI-powered layout understanding
- Expanded support for additional document formats and languages

Scalability Improvements:

- Distributed processing capabilities for enterprise-scale document volumes
- Cloud-native architecture for improved performance and accessibility
- Enhanced caching mechanisms for frequently accessed documents
- API development for integration with existing business intelligence systems

Project Duration: 3 months development cycle

Technology Stack: Python, Fast-API, Streamlit, OpenAI API, FAISS, Hugging Face Transformers, specialized PDF processing libraries

System Status: Fully functional prototype with comprehensive feature set meeting all specified requirements