Report

AI-POWERED SALES DECK GENERATOR

1.0 Introduction

In today's highly competitive and rapidly evolving business landscape, the quality of sales presentations significantly impacts an organization's ability to secure investment, drive growth, and effectively communicate its value proposition. Traditional pitch decks, primarily delivered as presentations, are often criticized for their lack of strategic depth and analytical precision. Current market solutions such as Pitch.com, Canva, and Google Slides emphasize creative designs but fall short in providing analytical depth and brand-specific tailored content crucial for effective investor engagement. A significant percentage of pitch decks fail to adequately address core investor and client queries related to competitive advantage, industry positioning, and detailed financial analytics, primarily due to generic or superficial content. This gap reveals a crucial market opportunity for solutions that seamlessly integrate data-driven insights with branding and market contexts.

In response to these industry limitations which aren't addressed in the market yet, we introduce DecGen.AI, an innovative AI-driven solution explicitly designed to automate and enhance the generation of sales decks with robust, data-driven content tailored precisely to individual brands.

Figure 1: Brand Mark of DecGen.AI - Empowering Brands with Automated, Data-Driven Sales Pitches

DecGen.AI's primary innovation lies in its ability to ingest a minimal initial input, including the brand name, description, colour scheme, logos, optional video assets, and essential marketing documents such as revenue statements and brochures, transforming these into a comprehensive, insightful, and visually cohesive web-based sales deck.

Unlike traditional tools that merely provide creative templates and superficial aesthetics, DecGen.AI leverages advanced Natural Language Processing (NLP) and Artificial Intelligence (AI) methodologies to analyse provided knowledge bases comprehensively. It autonomously synthesizes critical insights and analytics about the given brand, identifies industry-specific competitive advantages, and clearly communicates the strategic positioning of the brand within its market segment.

1.1 Key Implementation and Their Novelty

Automated Content Structuring from Knowledge Bases

Unlike competitors, DecGen.AI ingests structured and unstructured data from uploaded brand documents (e.g., brochures, financial reports) and intelligently generates context-aware content sections. This eliminates the manual effort of outlining a pitch deck, while ensuring the deck is grounded in factual brand narratives.

Dynamic Brand Integration

While tools like Canva and Pitch allow custom branding, DecGen.AI automatically adopts the brand's primary and secondary colour schemes, logos, and tonality across all generated sections without requiring manual theme selection. This approach ensures consistency and identity alignment. While future iterations could incorporate expanded customization features, our current focus prioritizes content intelligence and automation. By doing so, we ensure that foundational content generation is robust, and reliable customization can be seamlessly layered on in future versions.

Competitor-Aware Deck Generation

DecGen.AI conducts automated research on market competitors using the given knowledge base and positions the brand advantageously within the sales deck. Unlike traditional tools that rely on user input for competitor data, our system synthesizes publicly available insights and custom analytics to highlight a brand's edge. This builds investor confidence and supports strategic storytelling.

Interactable Data Visualizations

Based on the documents provided in the knowledge base, DecGen.AI extracts quantitative metrics using a multi-agent NLP pipeline. These metrics are then visualized using interactable, web-optimized charts and graphs, offering stakeholders intuitive, real-time views into company performance and projections. Unlike traditional static visualizations, this novel approach dynamically builds visuals from unstructured and structured financial documents, enhancing both comprehension and presentation value.

Sensitive Document Handling with Local Processing

A higher percentage of executives cite AI transparency and data privacy as top concerns, especially in finance. Recognizing the confidentiality concerns of finance professionals and enterprises, DecGen.AI introduces a unique document-tagging system. Any document in the knowledge base can be marked as 'sensitive,' triggering a locally deployed model to process it. This local evaluation strictly handles inference without training on the data or storing any traces. This privacy-first feature is unprecedented among pitch deck tools and ensures compliance with security expectations. In future production environments, this approach can be deployed in secure cloud containers or private servers, ensuring maximum control over sensitive corporate assets.

2.0 Methodology

2.1 Technologies Implemented

DecGen.AI is built using microservices-based architecture comprising three main layers: a React-based frontend, a Node.js + Express API backend, and Python Flask-based local AI services. Each layer is independently deployable and optimized for its specific role in the platform. Below are the core technologies used along with their respective purposes and the reasons behind their selection.

Front-End: React + JSX

- **Component Modularity**: Core UI features (e.g., `CreateDeck`, `DeckCharts`, `CompetitorAnalysis`) are built as reusable, testable components.
- **State Management**: React Context API ensures seamless and scalable state sharing across dynamic views.
- **Visualization Support**: Integration with leading charting libraries (e.g., Chart.js) enables interactive, data-driven storytelling.
- **Async UX**: Built-in React hooks (e.g., `useEffect`, `useState`) handle asynchronous tasks, such as document upload progress and API responses.

Back-End: Node.js + Express.js

- **Unified Stack**: JavaScript across both frontend and backend simplifies development and enhances code reuse.
- **Non-blocking I/O**: Efficiently handles concurrent file uploads and communication with AI services.
- **OpenAI API Integration:** Connects with OpenAI's GPT-4.1 model via secured REST endpoints to perform advanced NLP tasks such as content generation, brand tone adaptation, and sales narrative drafting. This centralized handling ensures fast and reliable inference directly within the backend pipeline.
- **Middleware Pattern**: Modular setup with middleware for authentication, CORS, logging, and file handling, improving maintainability.

Database: MongoDB

- **Flexible Document Store**: Ideal for storing unstructured data like deck sections, user annotations, and uploaded documents.
- **Schema Agility**: Adapts easily to evolving data structures and AI-generated metadata.
- **JSON-Native**: Enables smooth data interchange with JavaScript-based front-end and backend.

AI Microservice Layer: Python + Flask

- **Scalable Isolation**: Each AI task (e.g., NLP analysis, data extraction) is encapsulated within a stateless microservice.
- **Python-Native Libraries**: Leverages Hugging Face Transformers and advanced tokenization tools unavailable in Node.js.
- **Memory Efficiency**: Python's native memory management ensures high-throughput inferencing, especially when handling large documents.
- **Secure Integration**: RESTful Flask endpoints communicate securely with the Node.js backend, enabling asynchronous AI inference without coupling layers.

2.2 Dataset Utilized

Throughout the development of DecGen.AI, several datasets were employed to ensure robust model performance, particularly for financial named entity recognition (NER) and data extraction tasks. Below is a comprehensive list along with their specific usage and outcomes:

Financial-NER-NLP (Josephgflowers/Kaggle) - This specialized dataset was initially employed to fine-tune local models specifically for recognizing financial entities such as companies, financial metrics, and market-specific terminology. The tailored financial annotations enabled our models to more accurately interpret sensitive and precise financial data, essential for automating sales deck content creation.

Conll2003 (General NER with HuggingFace Dataset) - Utilized as a baseline dataset for named entity recognition tasks, aiding general-purpose extraction and understanding of entities across various contexts. It served as foundational training data to build upon more specialized financial datasets.

Custom-Collected Dataset (NewsAPI, RSS Selective News API, Alpha Vantage API) - A unique dataset was created through targeted data collection to enrich model training with both real-time and historical financial insights:

- **NewsAPI**: Used to gather recent market news and related corporate announcements (up to 250 latest articles), enhancing the AI model's contextual understanding and ability to generate timely, informed content.

- **RSS Selective News & Alpha Vantage API**: Integrated to extract real-time and historical financial metrics (such as stock prices, revenue figures, earnings reports) crucial for training the model to recognize and correctly contextualize financial indicators within generated decks.

While leveraging these datasets improved the model's overall performance, especially in financial contexts, certain limitations and challenges were encountered like local model deployments trained using these datasets occasionally exhibited hallucinations or inaccuracies, particularly when handling highly nuanced or specialised financial terms.

2.3 Overview of the System Workflow

This section provides a high-level summary of the DecGen.AI system architecture. A full, detailed explanation covering model configurations, processing flows, and logic is available in **Appendix A.**

The DecGen.AI system begins with user authentication through a React frontend hosted on port 3000. The login interface, powered by Login.jsx, sends user credentials to an Express.js backend on port 3001, which authenticates them against a MongoDB database. Upon successful login, a secure session token is generated, granting access to the protected dashboard rendered by Dashboard.jsx, with navigation managed via Sidebar.jsx.Once inside, users interact with FileManager.jsx for document uploads and CreateDeck.jsx for initiating sales deck generation. The file upload process is managed using the Multer middleware, supporting a variety of file types including PDFs, markdown, and video files. During upload, users must classify files as sensitive or non-sensitive which is a critical step that is stored in the MongoDB schema for security-preserving processing.

In CreateDeck.jsx, users enter brand-specific details including name, description, and positioning. They can select logos and media files already uploaded, and real-time validation ensures completeness of the form. Upon submission, a POST request is made to /api/decks, triggering the backend to fetch relevant files and initiate the AI processing pipeline based on document sensitivity.

Documents marked as sensitive are processed entirely locally through a Flask-based AI service (app.py) operating on port 5001. This service uses three Hugging Face models: LLaMA-3.1 for text generation, BART for summarization, and RoBERTa for sentiment/context extraction. These models extract document insights including keywords, summaries, and business metrics, ensuring no external API calls are made to preserve data privacy.

In contrast, non-sensitive documents are processed via a cloud-based multi-agent system. Text from all non-sensitive files is extracted and passed to MultiAgentDataExtractor.js, which spawns four GPT-4.1 powered agents via openai-wrapper.js. Each agent focuses on a distinct task: structural data extraction, business metric identification, temporal trend analysis, and competitive insights. Results are returned in structured JSON formats, validated locally, and integrated into a consolidated dataset.

A JavaScript-based validation pipeline checks table structures, verifies metric formats, and ensures time series consistency. Visual content is generated using generateChartConfigs, which selects optimal chart types (e.g., bar, pie, KPI dashboards) based on data context and richness. Sensitive and non-sensitive insights are merged without leaking sensitive content.

Finally, a templated HTML sales deck is assembled using Chart.js for interactive charts, brand summaries, and linked source documents. The React frontend renders these results using DeckCharts.jsx and DataChart.jsx, providing tooltips, data drilldowns, and security status indicators.

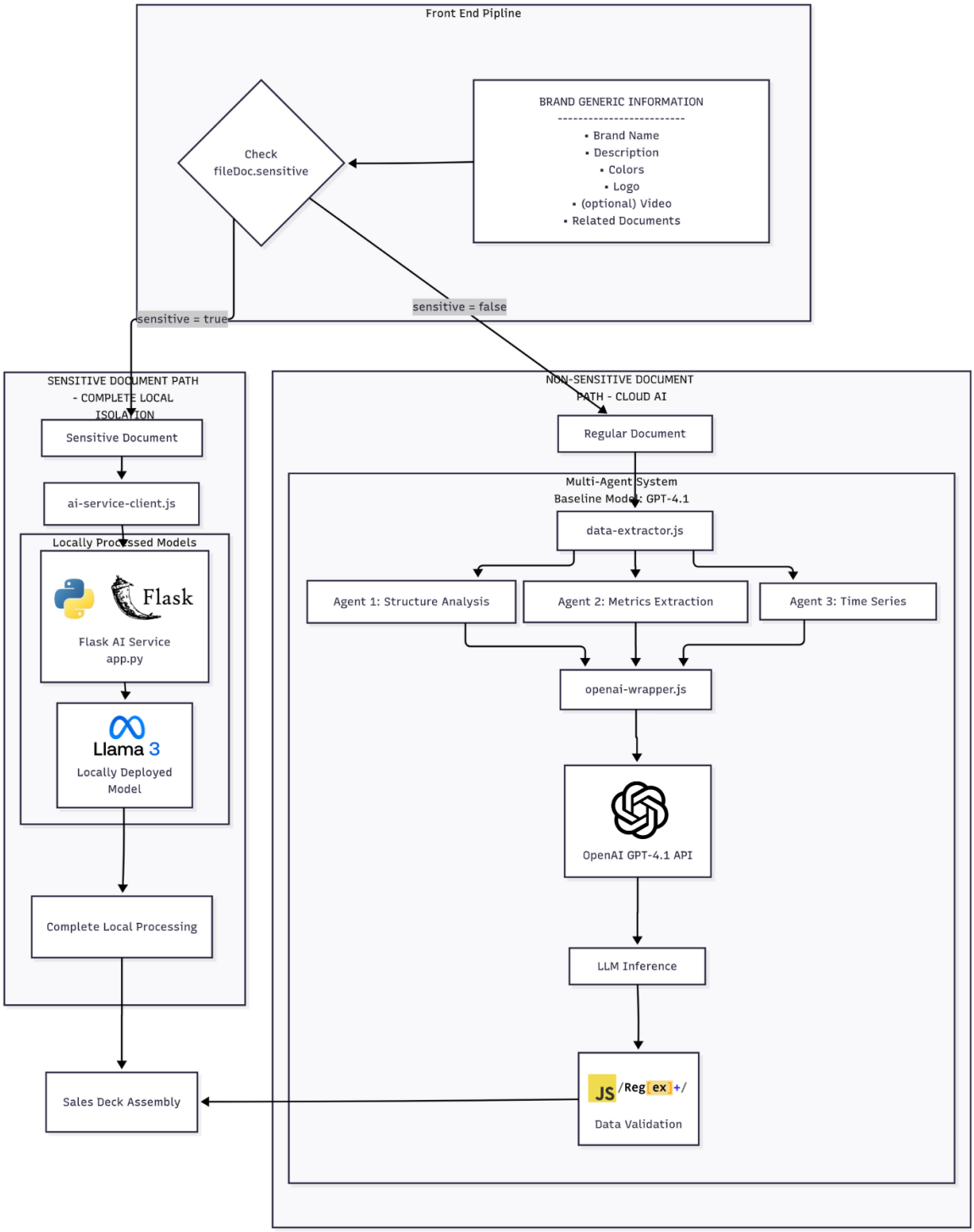

Figure 2 DecGen.AI System Architecture Overview

3.0 Experimental Results

The experimental phase of DecGen.AI focused on systematically evaluating the effectiveness and accuracy of the implemented models and processes, comparing them against alternative solutions, and improving the overall user experience through enhanced visualization and strategic analysis.

3.1 Evaluation Methods

To rigorously evaluate our system, we adopted two distinct evaluation strategies:

1. **Sensitive Document Processing (Local LLaMA Model)**
    - **Feasibility Testing**: Conducted using an 8B-parameter LLaMA model due to its balance of performance and efficiency, ideal for local deployments and sensitive data processing.
    - Evaluated the model on financial statements, specifically checking its ability to accurately identify financial entities (e.g., monetary values, fiscal metrics).
2. **General Document Processing (GPT-4.1 Multi-Agent)**
    - Focused primarily on output quality at each processing stage, measuring content relevance, completeness, clarity, and visual engagement.

3.2 Sensitive Document Model Evaluation

We conducted comprehensive tests using the LLaMA 8B model:

Performance Metric	Average Quality Score	Average Processing Time
Masking	9.2/10	9.36 seconds
Remapping	8.5/10	17.12 seconds

These strong initial results justified further fine-tuning of this model using targeted on freshly collected financial news and statistical datasets.

3.3 Dataset Collection and Model Fine-tuning

Initially, a smaller 3B-parameter model was trained using publicly available datasets (Josephgflowers/Kaggle, HuggingFace/Conll2003), but it faced significant hallucination issues due to dataset limitations and insufficient parameters. Consequently, we fine-tuned the proven 8B-parameter LLaMA model using financial-specific data collected from NewsAPI, RSS News Feeds (Full Content Financial News) and Alpha Vantage API (Market news). A JSON-based dataset with over 90 entries was created and thoroughly preprocessed for optimal performance. **(Detailed dataset preprocessing steps and analysis, including relevant screenshots, are available in Appendix B.)**
The fine-tuned LLaMA model was successfully deployed on the Hugging Face Hub, enhancing ease of integration and scalability in the application.

3.4 General Document Processing and Multi-Agent Approach

The general documents pipeline initially relied solely on GPT-4.1 for analytics and visualization, resulting in basic, static outputs. Recognizing this limitation, we implemented a sophisticated multi-agent system:

- **Agent 1 (Structure Analysis)**: Identified structured data like tables and financial metrics.
- **Agent 2 (Metrics Extraction)**: Precisely captured quantitative data, KPIs, and financial ratios.
- **Agent 3 (Time Series Analysis)**: Analyzed historical data, seasonal trends, and forecasts.
- **Local Data Validator**: Ensured extracted data integrity and coherence before visualization.

This multi-agent framework substantially improved the accuracy and completeness of data extraction. It enabled dynamic and interactive data visualizations powered by Chart.js, significantly enhancing the final sales deck's appeal and interactivity. (Refer to Appendix A for detailed implementation workflows, agent interactions, and architecture diagrams.)

3.5 Competitive Intelligence and Strategic Analysis

To improve market analysis accuracy, we employed advanced structural prompting techniques alongside careful prompt engineering. These methods effectively guided GPT-4.1 to autonomously perform detailed competitor analysis, identify strategic positioning, highlight strengths, and propose actionable insights tailored precisely to each industry. The autonomously generated strategic recommendations consistently demonstrated practical value, reinforcing DecGen.AI's viability and applicability in real-world scenarios. (Relevant screenshots showing strategic analysis outputs and comparisons can be found in Appendix B.)

3.6 Comparative Performance

Compared to existing solutions that focus solely on visual appeal or basic content generation, DecGen.AI demonstrated notable improvements in:

- Accuracy of financial data extraction
- Quality and depth of competitor analysis
- Interactive and engaging data visualizations

This comprehensive evaluation clearly positions DecGen.AI as an innovative, effective, and strategically valuable solution for automated sales deck creation.

4.0 Contribution Breakdown

Team Member	Key Contributions
Karthik Narayan	• Proposed the overall project plan, architecture, and structure • Set up and managed a collaborative GitHub environment with structured commits • Developed the initial system and integrated frontend, backend, and AI components • Handled UI/UX and logic implementation (File Manager, processing workflows, etc.) • Handle advanced functionalities of the system • Conducted first-phase evaluation of AI models • Final system testing, refinement, and assisted with deployment and reporting
Gishor Thavakumar	• Designed high-level architectural diagrams and system flow • Led data collection, preprocessing, and dataset structuring • Proposed and built the Sensitive Document Handling Pipeline • Experimented and trained models (multi-agent systems and LLaMA variants) • Improved output pipelines, added interactive charts • Deployed models and datasets to Hugging Face Hub – Worked all documentation, the final report, and presentation

5.0 Link to Deployed Resources

GitHub Link: https://github.com/knarayan9829/AI-Sales-Deck-Generator

Dataset: https://huggingface.co/datasets/tgishor/financial-training-dataset

Model: https://huggingface.co/tgishor/financial-llama-8b-experimental

Full screenshots demonstrating key features, workflows, and competitive analysis outputs are provided in **Appendix C** for visual reference and evaluation.

Appendix A - Detailed System Workflow

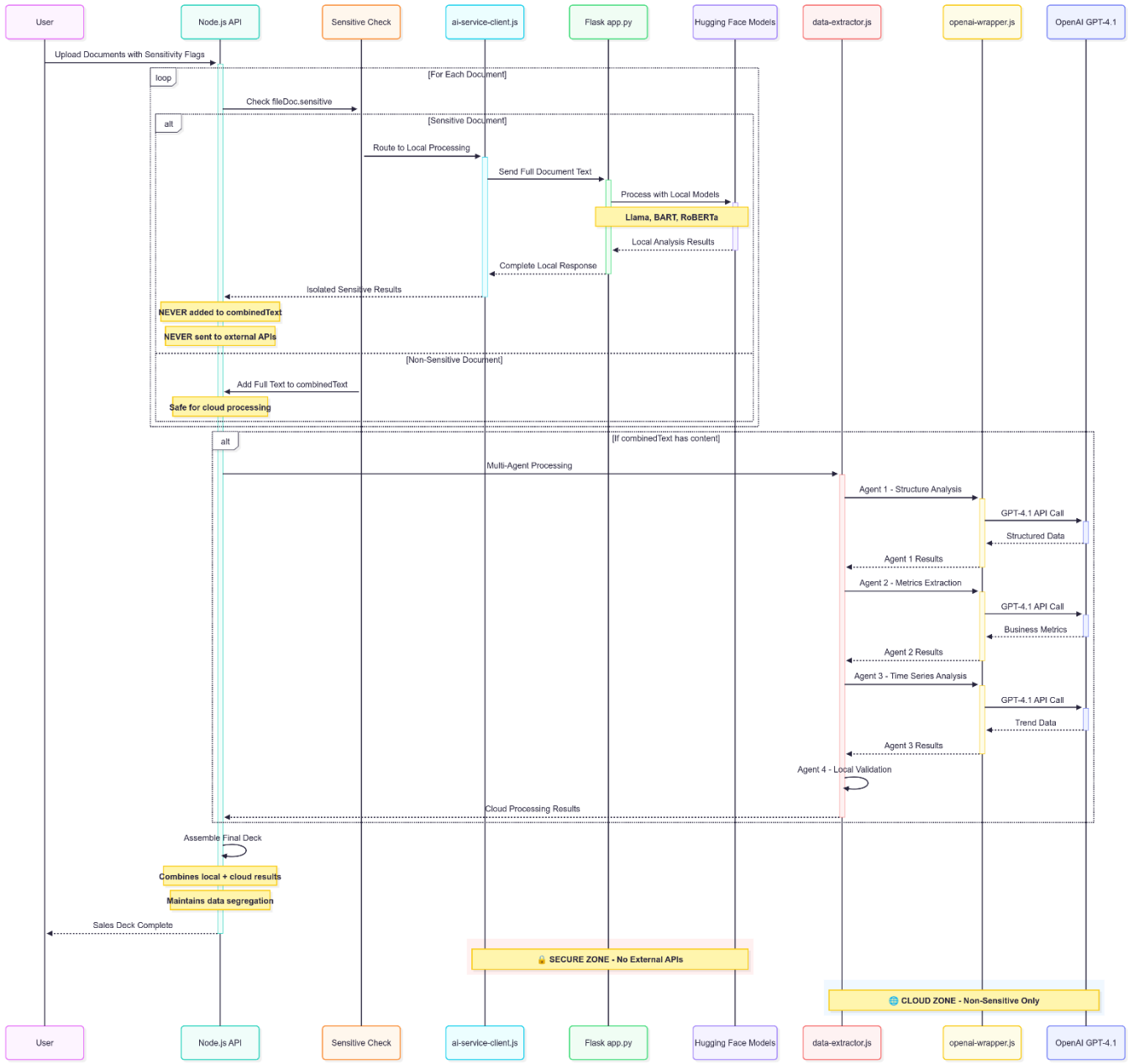

Figure A.1 – End-to-End Document Processing Sequence Flow in DecGen.AI. This sequence diagram outlines the step-by-step execution flow of DecGen.AI from user document upload to final sales deck assembly.

User Authentication and System Entry – The system initiates with user authentication through a React-based frontend application hosted on port 3000. Users access the login interface rendered by the Login.jsx component, which presents authentication fields for username and password validation. Upon credential submission, the frontend transmits authentication requests to the Node.js Express backend server operating on port 3001. The authentication middleware validates user credentials against the MongoDB database. Successful authentication establishes a secure session token, enabling access to the protected dashboard interface while unauthorized users receive rejection responses with appropriate error messaging.

Dashboard Navigation and File Management – Upon successful authentication, users navigate to the Dashboard.jsx component, which renders the main application interface with sidebar navigation provided by Sidebar.jsx. The dashboard presents multiple operational modules, with primary focus on the FileManager.jsx component for document upload operations and CreateDeck.jsx for sales deck generation. The file management interface utilizes Multer middleware on the backend for handling multipart form data uploads, supporting multiple file formats including PDF, TXT, MD, and various video formats. The system implements a sophisticated sensitivity classification dialog that

prompts users to mark documents as sensitive or non-sensitive during the upload process, storing this critical security metadata in the File schema within MongoDB.

Brand Information Collection and Validation – The sales deck process starts in the CreateDeck.jsx component, which presents a structured form for collecting brand details. Users provide a brand name (used for deck identification and URL), a short description (for elevator pitch content), and positioning details. Logos and videos are selected from previously uploaded media, while the knowledge base section displays uploaded documents clearly marking sensitive files with security badges. Real-time validation ensures all required fields are correctly filled before submission.

Document Sensitivity Routing and Processing Initiation – After form submission, a POST request is sent to */api/decks* with brand metadata, selected file IDs, and a *hasSensitiveFiles* flag. The backend retrieves the corresponding File documents from MongoDB and begins a classification loop. Each file's sensitive flag determines its processing route either secure local handling for sensitive content or external AI processing for standard documents. This branching mechanism forms the core of the system's privacy-preserving design, ensuring sensitive data never leaves the local environment.

Sensitive Document Local Processing Pipeline – Documents marked with sensitive flags undergo complete local processing through the ai-service-client.js module, which establishes communication with the Flask AI service operating on port 5001. The Flask application, implemented in app.py, loads three specialized Hugging Face transformer models: Llama-3.1-8B-Instruct for comprehensive text generation and analysis, BART-Large-CNN for advanced document summarization, and RoBERTa for sentiment analysis and contextual understanding. The local processing pipeline extracts the complete document text using the *extractTextFromFile* utility function, then transmits this content to the *processSensitiveDocument* endpoint. The Flask service processes documents through parallel model inference, generating keywords through Llama's advanced prompting capabilities, creating business-focused summaries via BART's encoder-decoder architecture, and extracting business metrics through pattern matching algorithms enhanced with Llama's analytical capabilities. The service generates strategic insights by combining extracted keywords and metrics through carefully crafted prompts, while plot data generation provides visualization suggestions based on identified data patterns. All processing occurs within the local computational environment with zero external API interactions, storing results in isolated memory structures that never interface with cloud services.

Non-Sensitive Document Multi-Agent Processing – Non-sensitive documents are processed via a secure cloud-based AI pipeline. Their text is extracted using *extractTextFromFile* and combined into a single *combinedText* variable. This corpus is then passed to the *MultiAgentDataExtractor* in data-extractor.js, which spawns four specialized agents each handling a distinct analytical task. All agents share a unified interface with OpenAI's GPT-4.1 through openai-wrapper.js, enabling coordinated and scalable cloud-based processing.

Multi-Agent System Architecture and Processing – The multi-agent coordinator initiates parallel processing across four specialized agents, each targeting distinct data extraction objectives. Agent 1 implements structure analysis functionality, crafting comprehensive prompts that instruct GPT-4.1 to identify tabular data, financial information, performance metrics tables, competitive comparison data, and market analysis tables from the complete document corpus. The agent transmits these prompts through openai-wrapper.js, which interfaces with OpenAI's GPT-4.1 model using the chat completions API with optimized temperature settings for analytical consistency. Agent 2 focuses exclusively on metrics extraction, employing sophisticated prompt engineering to identify quantifiable business metrics, key performance indicators, growth rates, customer satisfaction metrics, market share data, and financial ratios. Agent 3 specializes in temporal data analysis, extracting historical performance data, forecasting information, year-over-year comparisons, seasonal trends, and competitive performance tracking over time periods. Each cloud-processing agent receives JSON-structured responses from GPT-4.1, which undergo parsing and validation before integration into the consolidated data structure.

Local Data Validation – This process operates entirely within the local computational environment, performing comprehensive data validation and cleaning operations through JavaScript-based algorithms. This pipeline receives the combined output from all cloud-processing agents and implements rigorous validation protocols including table structure verification ensuring proper headers and row consistency, metrics validation confirming valid numerical values and appropriate units, and time series validation verifying temporal progression and data point sufficiency.

Content Generation and Competitive Intelligence - After data validation, the system triggers *generateChartConfigs* to translate extracted tables, metrics, and time series data into optimized visual formats. A chart selection algorithm evaluates data structure (e.g., row/column count, data types, business context) to assign appropriate visualizations such as pie charts for distributions, bar charts for comparisons, and KPI dashboards for metrics. A scoring system ranks charts

based on business relevance, data richness, competitive insight, and presentation impact. Financial and performance metrics are prioritized. The final output includes high-priority charts for immediate use and secondary charts for deeper analysis, balancing clarity with visual depth.

Final Assembly - In the final phase, results from both sensitive and non-sensitive pipelines are combined with strict segregation. Sensitive outputs remain isolated and contribute insights without exposing original content. Non-sensitive results including competitive analysis, summaries, and charts are assembled into a full HTML presentation using templated layouts and Chart.js for interactivity.

The final HTML deck includes:

- AI-generated brand overviews
- Chart dashboards (Chart.js)
- Strategic competitor insights
- Linked source documents (with security tagging)
- Processing transparency and audit trails (local vs cloud)

Response Generation and Frontend Integration - Upon completion of all processing phases, React frontend receives structured responses enabling immediate preview rendering through DeckCharts.jsx and DataChart.jsx components, interactive chart exploration with tooltips and data drilling capabilities, and processing status indicators documenting security compliance. This comprehensive technical workflow ensures secure, intelligent, and automated generation of professional sales presentations while maintaining security protocols for sensitive business information.

Appendix B - Dataset Preprocessing & Model Fine-Tuning Workflow

Dataset Collection Sources

The dataset was assembled from multiple financial data APIs, yielding roughly 90 JSON-formatted documents relevant to financial news and market data. We used NewsAPI for general business news, RSS News (Web Scrapping) for company-specific news, and Alpha Vantage for market data and headlines. NewsAPI provides RESTful JSON search results for current and historic news articles from over 150,000 global sources.

Dataset Preprocessing Pipeline

The raw JSON data was cleaned and standardized:

- Irrelevant fields (e.g., author name, images) were removed.
- Title, body, and summary were merged into a unified article field.
- Null values and duplicates were filtered out.
- Articles were formatted into prompt–completion pairs and tokenized for LLaMA input.

To better understand the structure and quality of the dataset, we also conducted basic data analytics. The insights ensured data consistency and guided adjustments in preprocessing. **(**See Figure B.1 – Financial Data Collection Analysis**)**
This included analysis of:

- Article type distribution
- Text length distribution
- Source frequency

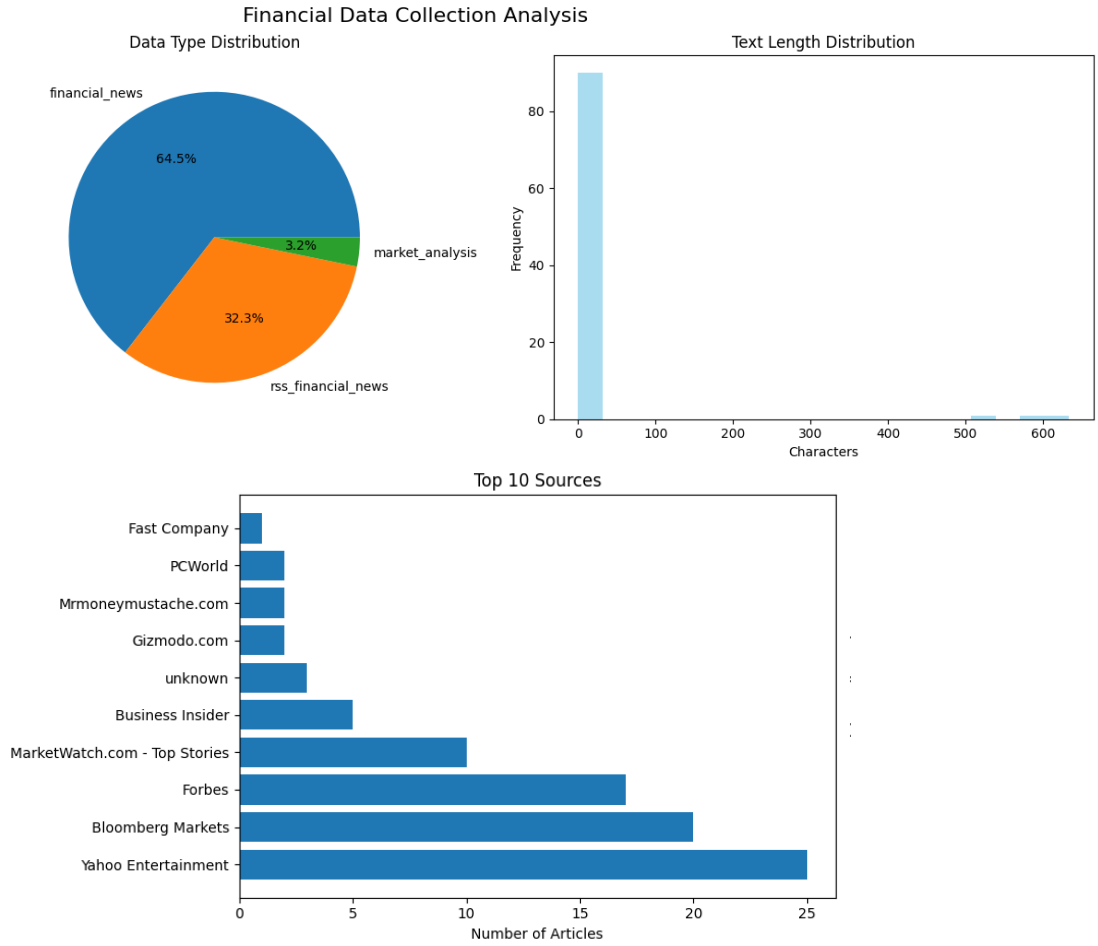

Figure B.1 – Financial Data Collection Analysis

Model Selection and Justification

An initial 3B-parameter model was trained using publicly available datasets; however, it exhibited frequent hallucinations and struggled with financial domain adaptation. The attempt was conducted using this Google Colab notebook (training attempt documented). Due to the model's limited parameter capacity and suboptimal alignment with the domain-specific data, it failed to produce reliable outputs.

As a result, we transitioned to the more capable LLaMA-8B model, which demonstrated significantly better contextual understanding and accuracy during feasibility testing, making it a more appropriate choice for handling sensitive financial documents.

Training and Finetuning Overview

Fine-tuning was carried out using a custom PyTorch training pipeline with the Hugging Face Transformers library. Instead of using the `TRL` SFTTrainer, we implemented a manual supervised instruction tuning setup with optional LoRA-based parameter-efficient training.

The LLaMA-8B model (`meta-llama/Llama-3.1-8B-Instruct`) was loaded via `AutoModelForCausalLM` and modified to include LoRA adapters on key linear layers (e.g., `q_proj`, `v_proj`, etc.). The model was trained on a JSON-based dataset of financial articles and summaries, formatted as instruction–completion pairs, with loss computed over the completion text only.

Key training configurations included:

- **Batch size**: 1 per device with gradient accumulation (effective batch ~16)
- **Sequence length**: Up to 1024 tokens
- **Optimizer**: AdamW with learning rate 1e-5 and warmup steps
- **Loss**: Cross-entropy on completion tokens

- **Precision**: fp16 disabled, bf16 used if available
- **LoRA setup**: `r=4`, `alpha=8`, `dropout=0.1` applied manually

Training proceeded smoothly on GPUs with ≥16GB VRAM. Loss decreased steadily across epochs. The final model and tokenizer were saved to `financial_llama_model_clean`, along with LoRA weights and a base checkpoint. The full training implementation, including configuration logic and manual LoRA integration, is available via Google Colab: LLaMA8B_Financial_LoRA_Training.ipynb

Deployment and Hugging Face Hosting

After fine-tuning, the model artifact was uploaded to the Hugging Face Hub as a private repository. This enabled immediate deployment via Hugging Face's inference infrastructure. In particular, we enabled the Hugging Face Inference API for the model, which provides a serverless, managed endpoint for running the model in production. The Inference API is a "fast way to get started" with a hosted service on HF's cloud, without needing to provision any servers. For production use, one could also spin up a dedicated Inference Endpoint (a paid, managed deployment) to ensure scalability and reduced latency. Hosting the model on Hugging Face provides several benefits: automatic load balancing and scaling, no server maintenance, version control, and usage logging.

Hugging Face Hosted Dataset: https://huggingface.co/datasets/tgishor/financial-training-dataset

Hugging Face Hosted Model: https://huggingface.co/tgishor/financial-llama-8b-experimental

Sample Evaluation

Post-deployment, the model was tested on unseen sensitive documents. The manual review confirmed the outputs were relevant, accurate, and hallucination-free, validating its suitability for real-world financial data scenarios.

Appendix C – Final System Evaluation and User Interface Overview

System Dashboard and Navigation

The main dashboard serves as the central hub of the DecGen.AI system, presenting a clean layout with clearly labeled modules. Key actions such as File Upload and Create Deck are featured prominently, and the sidebar navigation provides quick access to these and other tools. Consistent iconography and a responsive design ensure that users can easily identify their current section at a glance. Additionally, when the system is executed, all processing activities are clearly documented through structured log files, which are separated into distinct functional sections and enhanced with interactive visual cues (e.g., emojis) to improve traceability and user clarity during debugging or runtime observation.

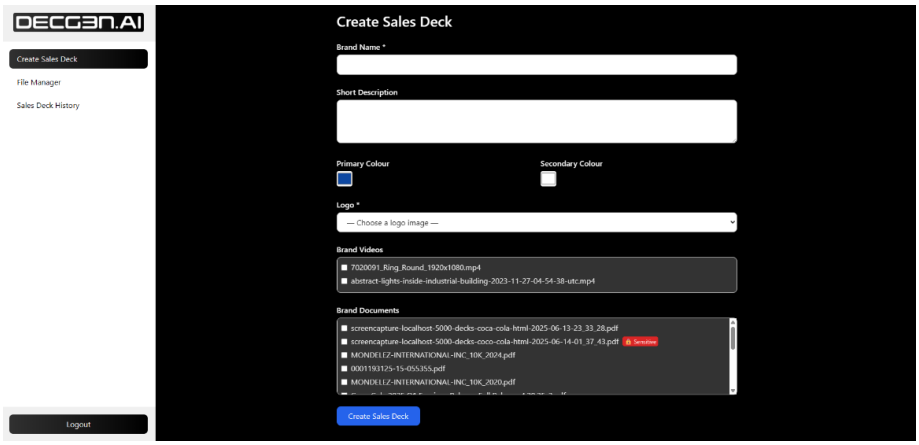

Figure C.1 – A screenshot of the home dashboard with the sidebar navigation

Document Upload and Sensitivity Classification

The document upload interface allows users to select or drag-and-drop common formats (e.g. PDF, DOCX, TXT) for import. After upload, each document is automatically analysed and tagged with a sensitivity label (such as “Public” or “Confidential”) based on its content. This mechanism follows standard data classification practices, mapping content to categories (such as confidential, sensitive, or public) used for security and governance. In the FileManager view, each file is listed with a visual badge indicating its classified sensitivity, allowing users to quickly scan which files are flagged as sensitive.

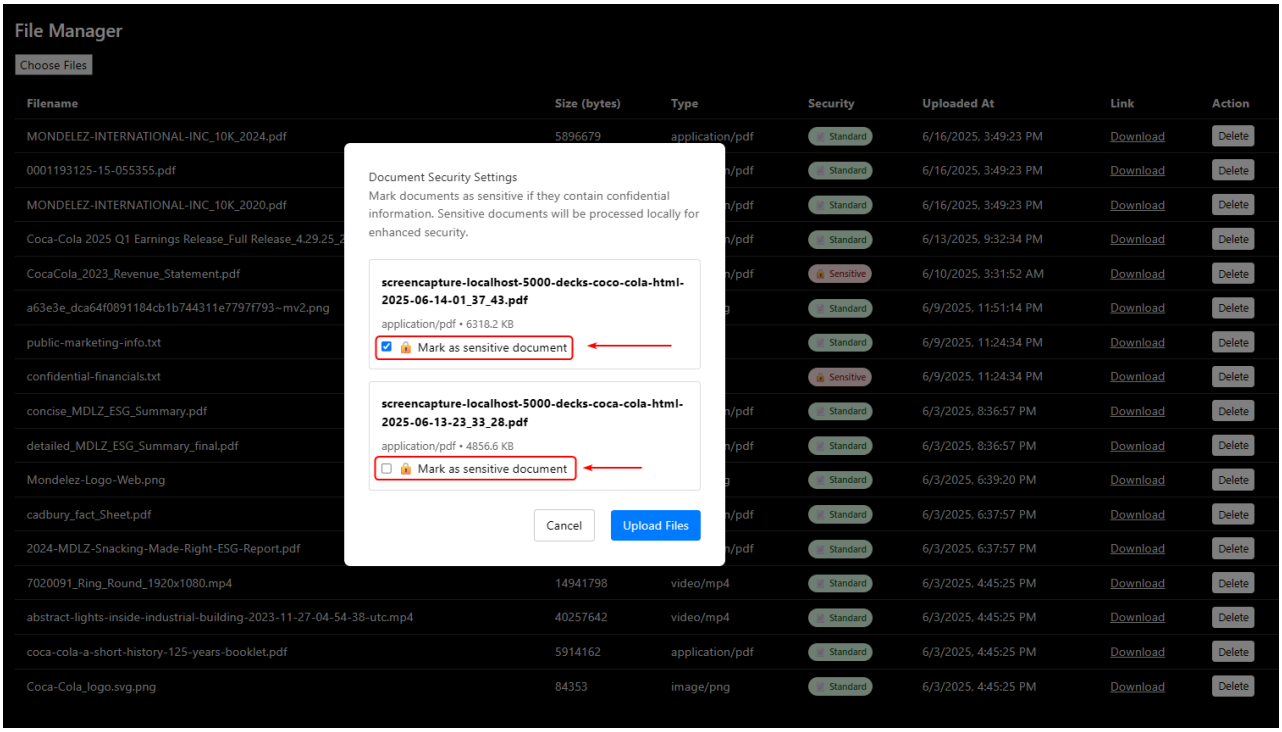

Figure C.2 – Tagging Documents as Sensitive During Uploading into the File Manager

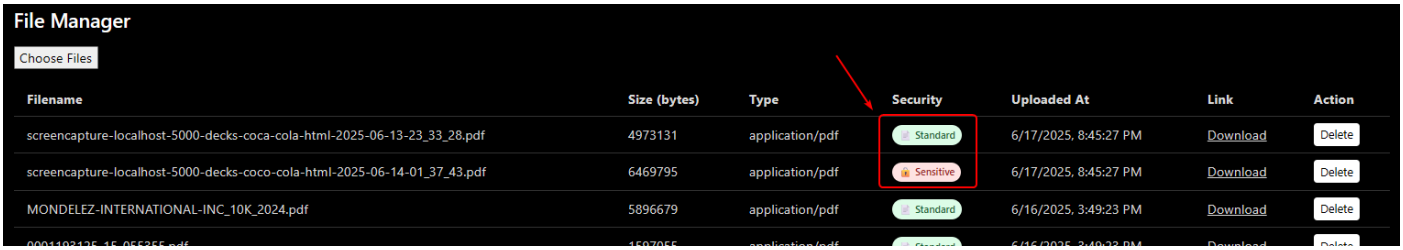

Figure C.3 – A screenshot of the FileManager interface with visible classification badges

Brand Creation Workflow

The **Create Deck** workflow guides users through entering brand and campaign details before generating a sales deck. The workflow typically includes steps such as:

1. **Metadata Input:** Enter brand information (name, tagline, target audience, etc.).
2. **Media Linking:** Upload or link media assets (logos, images, videos) associated with the brand.
3. **Document Selection:** Choose which of the uploaded documents to include as content sources.

Each step provides real-time validation of user inputs, giving immediate feedback (e.g. checkmarks or error hints) so that format or completeness issues can be corrected instantly. This live validation approach boosts user confidence and sense of progress, as noted in UX research.

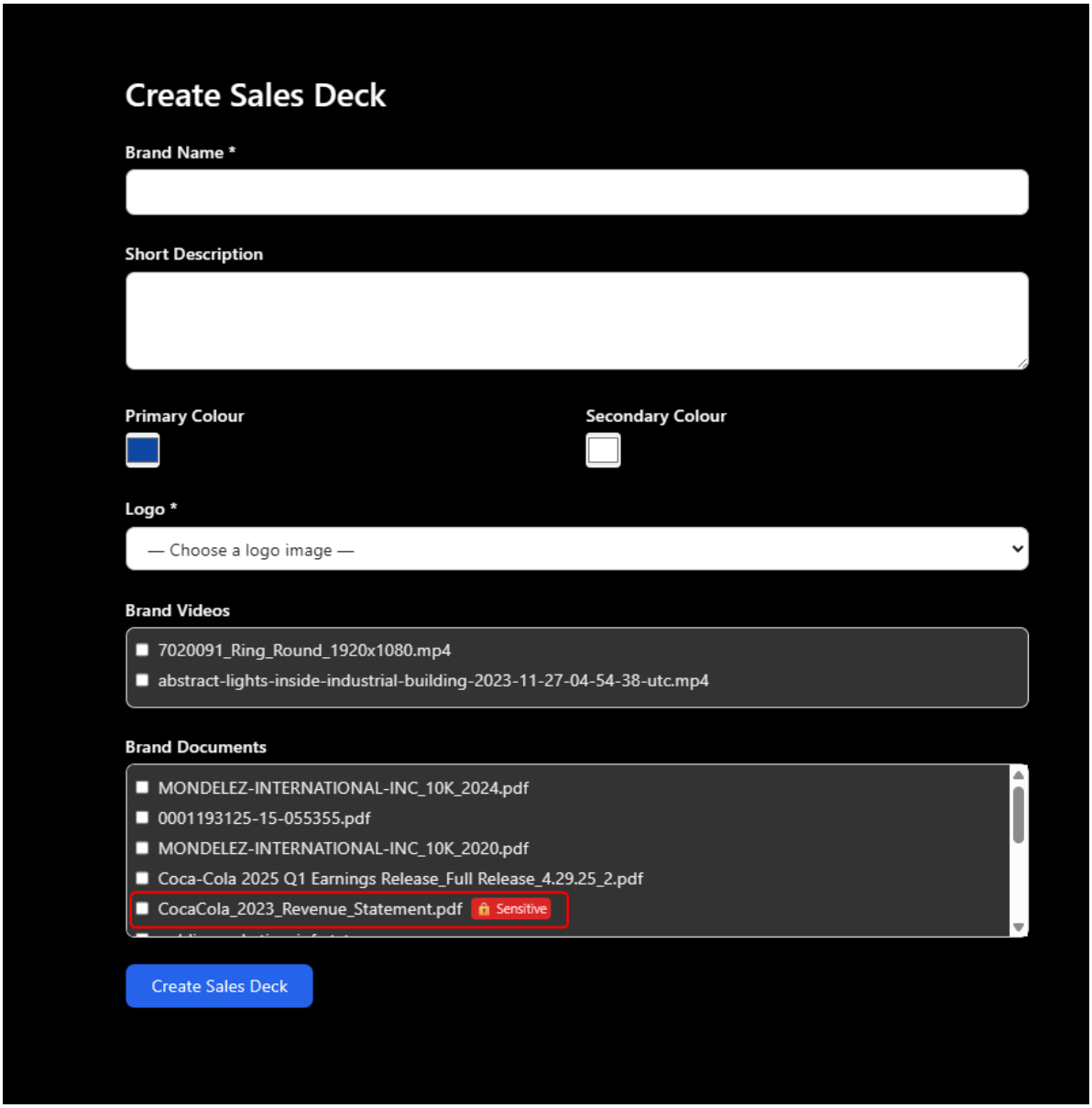

Figure C.4 – A screenshot of the brand form interface (showing input fields and inline validation indicators) illustrating the above explanation

Sales Deck Generation Phases

After submission, the system routes the request into one of two processing pipelines based on document sensitivity. Sensitive documents are handled in an isolated local pipeline (e.g. using a locally hosted LLM) to ensure privacy, whereas non-sensitive documents enter a cloud-based multi-agent processing pipeline. In the non-sensitive pipeline, a base orchestrator agent (powered by GPT-4.1) coordinates multiple specialized sub-agents (e.g. one agent parses document structure, another extracts key metrics, a third analyzes time-series data). The agents operate in parallel and return their results to the orchestrator, which then assembles an integrated sales deck. This multi-agent design is motivated by the fact that breaking complex tasks into specialized agents can improve performance and scalability. Research indicates that iterative, collaborative AI workflows often yield higher-quality outputs than single-pass generation.

The final sales deck incorporates interactive charts to display key performance indicators (KPIs) and trends derived from the data. These charts are rendered using Chart.js, a popular HTML5 charting library chartjs.org. Chart.js's canvas-based rendering makes it very performant even with large datasets. In practice, metrics and forecasts output by the AI agents are fed into Chart.js components (such as bar, line, and pie charts) to visualize trends and comparisons.

Figure C.5 – A screenshot before integrating multi agent approach

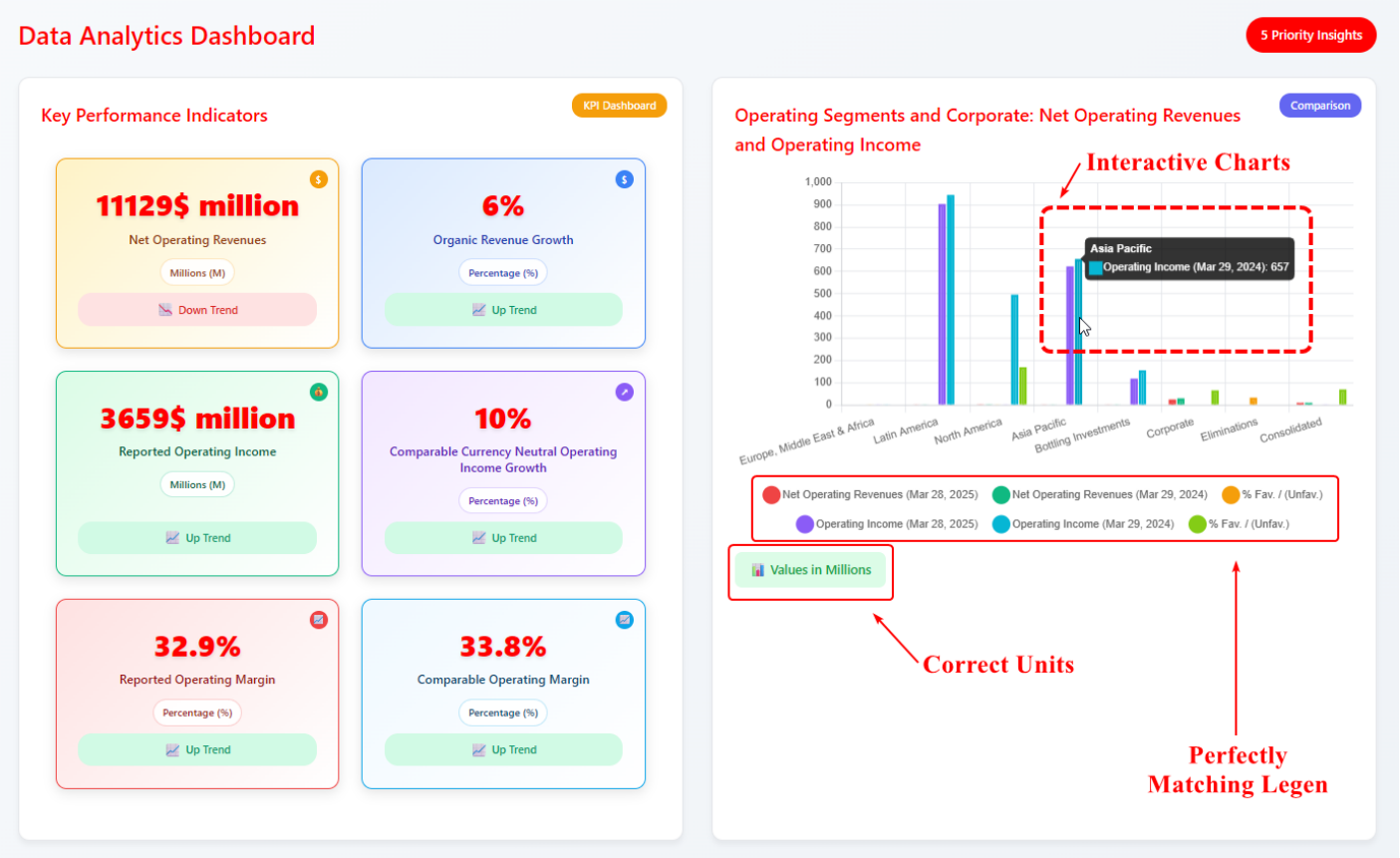

Figure C.6 – Results after integrating multi agent approach to extract metrics and display visualization. Only few is represented in this snippet

Competitive Analysis Output

For competitive positioning, one agent generates structured insights by comparing the brand to market benchmarks. The raw output is produced in JSON form, which the system then transforms into formatted HTML for the final deck (e.g. converting a JSON array of competitor attributes into an HTML table). This JSON-to-HTML conversion allows the insights (such as competitor strengths, weaknesses, and opportunities) to be presented in a clear, cards layout.

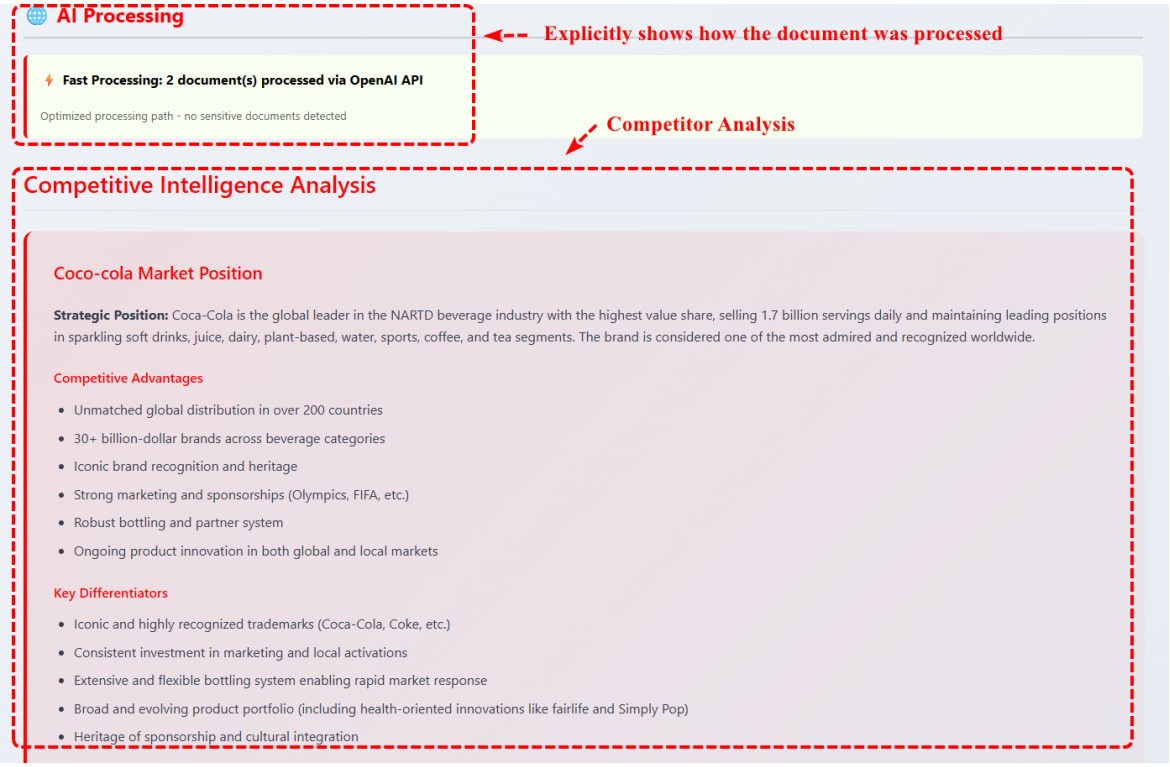

Figure C.6 – Market Position Summary with AI Processing Indicators

The screenshot below demonstrates automatically generated competitor profiles categorizing direct and indirect rivals. Each card highlights strengths and challenges, structured with consistent visual formatting to help comparative market analysis.

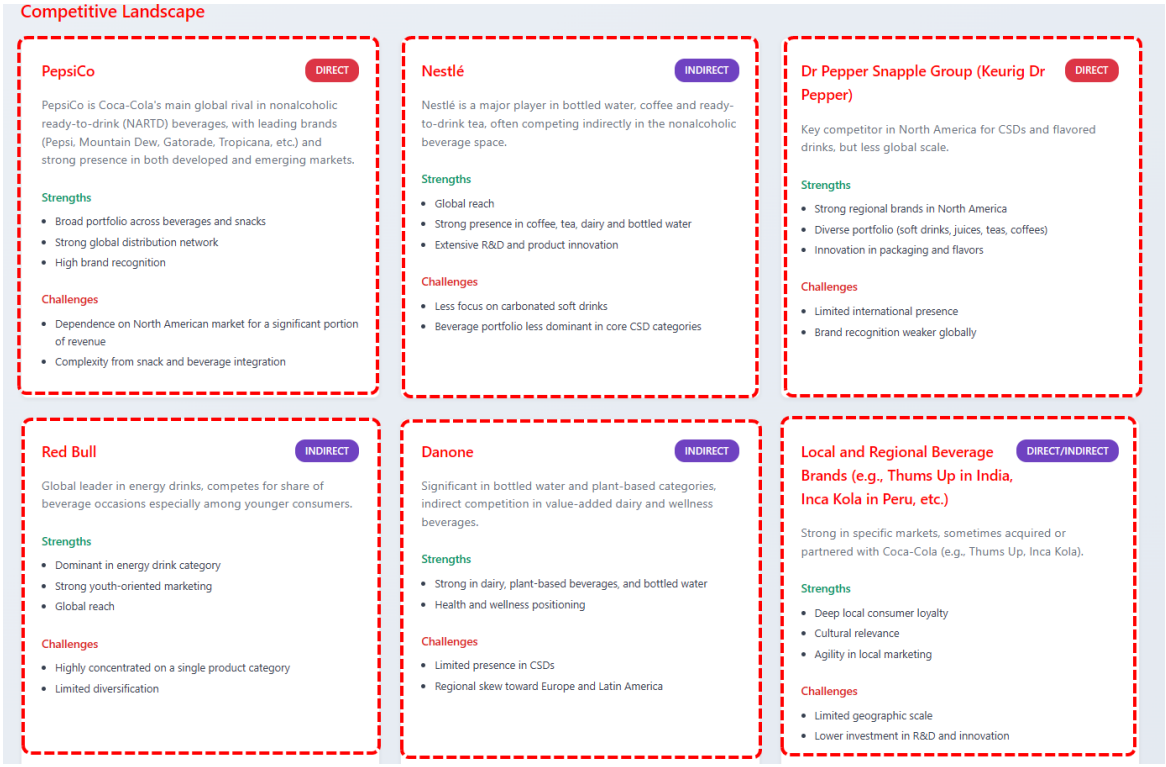

Figure C.7 – Structured Competitor Intelligence Cards Rendered in the Sales Deck

The snippet below shows the comprehensive market narrative generated through the GPT-4.1 system. It includes contextual analysis of market trends, growth projections, and brand's (in this case Coca-Cola's) innovation and community initiatives.

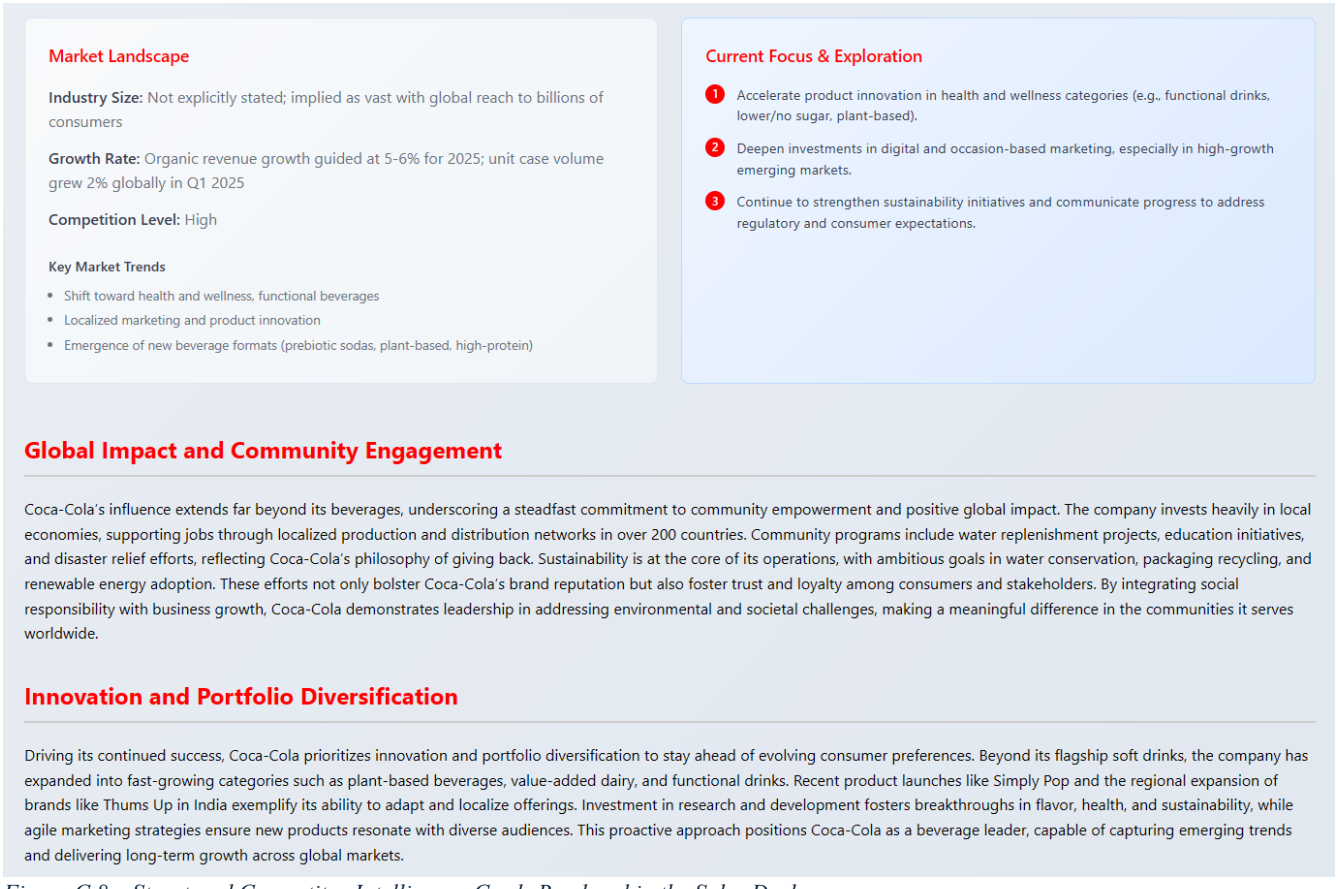

Figure C.8 – Structured Competitor Intelligence Cards Rendered in the Sales Deck