

Automation of Slide Deck Generation and Banking Control Assessment

Realization document

Fabrice Elono Piseth
Student Bachelor Applied Computer Science

Table of Contents

1. INTRODUCTION TO KPMG BELGIUM	3
1.1. Presence and offices	3
1.2. Core Values	4
1.3. Clients and Industries	5
1.4. KPMG Lighthouse Belgium	5
2. INTERNSHIP ASSIGNMENT	7
2.1. Slide Deck Generator	7
2.2. Banking Control Assessment	7
3. ANALYSIS	8
3.1. Tools Used – Overview and Purpose	8
3.2. Vector Store Comparison: FAISS vs Alternatives	10
3.3. Why FAISS Was Selected	10
3.4. Langflow vs Streamlit + Langchain	11
4. PROJECT TIMELINE	12
5. PROJECT 1 : SLIDE DECK GENERATOR	13
5.1. Use Case and Motivation	13
5.2. Tools and Technologies Used	14
5.3. Architecture and Design	15
5.4. Implementation Highlights	20
5.5. Langflow–Streamlit Integration	22
5.6. Results and Outcomes	23
6. PROJECT 2 :BANKING CONTROL ASSESSMENT AUTOMATION	24
6.1. Use Case & Motivation	24
6.2. Context and Business Value	25
6.3. Tools & Technologies	26
6.4. Architecture & Implementation	27
6.5. Why Cosine Similarity?	33
6.6. Streamlit Integration & Final Outcome (Duplicate Control)	34
7. CONCLUSION	38
8. REFERENCE LIST	39
9. BIBLIOGRAPHY	39
10. ATTACHEMENTS	40

1. Introduction to KPMG Belgium

KPMG Belgium is a member firm of the global KPMG network, which provides audit, tax, legal, and advisory services in 143 countries. With more than 1,900 professionals across Belgium, the firm combines deep local expertise with global insights to help clients navigate their most complex challenges. KPMG Belgium is known for delivering high-quality, multidisciplinary services across sectors including finance, public sector, technology, life sciences, energy, and more.

1.1. Presence and offices

The firm operates in over eight locations in Belgium, strategically spread to remain close to its clients:

Brussels (Head Office) , Antwerp , Ghent , Hasselt , Liège , Kortrijk , Doornik , Louvain-La-Neuve , Mont-Saint-Guibert . This geographic distribution ensures accessibility and localized support across Flanders, Wallonia, and Brussels.



Figure 1 : Map indicating the different offices of KPMG Belgium

1.2. Core Values

KPMG Belgium operates under a strong ethical framework guided by five core values:

- **Integrity** – Doing what is right, always.
- **Excellence** – Continuously learning and improving.
- **Courage** – Acting with confidence and challenging constructively.
- **Together** – Working collaboratively across boundaries.
- **For Better** – Striving to make a positive impact on clients, people, and society.

These values are embedded in the firm's day-to-day operations and its long-term strategic goals.

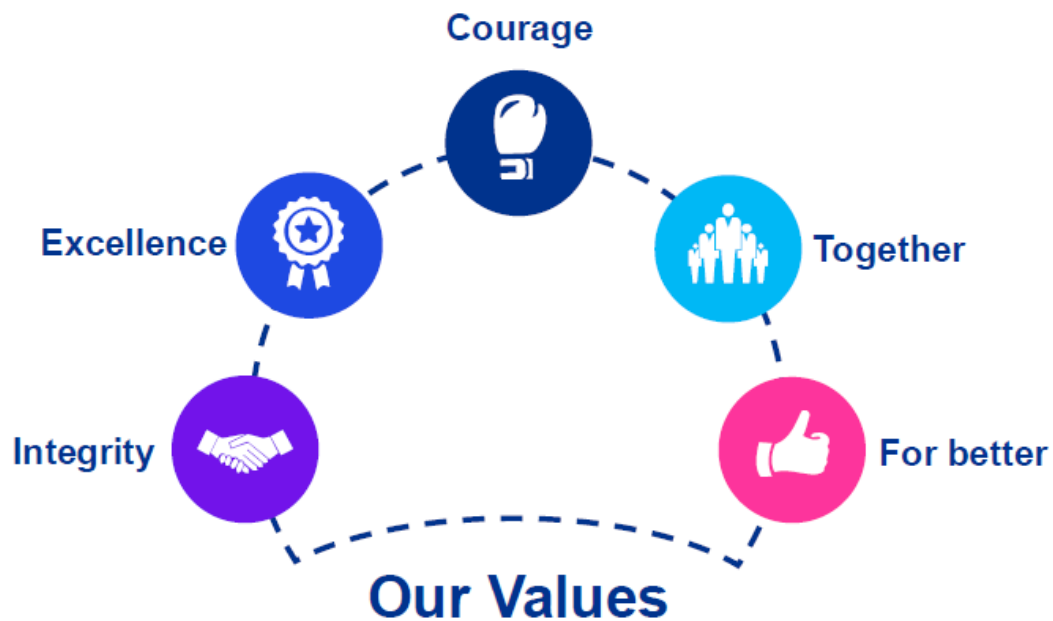


Figure 2 : Core Values at KPMG Belgium

1.3. Clients and Industries

KPMG Belgium supports a wide portfolio of clients across both private and public sectors. These include:

- **Financial Services** (banking, insurance)
- **Energy & Infrastructure**
- **Consumer and Retail**
- **Life Sciences**
- **Government and Public Sector**
- **Technology, Media, and Telecom (TMT)**
- **Family-Owned Businesses**
- **Real Estate and Automotive**

This sector-driven structure allows KPMG teams to offer highly specialized, context-aware advice tailored to the challenges of each industry.

1.4. KPMG Lighthouse Belgium

KPMG Lighthouse is KPMG's center of excellence for data, AI, and emerging technologies. In Belgium, it brings together experts across five specialized domains to help clients drive digital transformation and unlock value from their data.

Core Departments and Their Focus

- **Data Strategy & Management**
Focuses on building trusted, governed data foundations and helping organizations treat data as a strategic asset.
- **BI & Analytics**
Delivers dashboards, KPIs, and data visualizations to support smarter business decisions.
- **Advanced Analytics & Machine Learning**
Applies machine learning and AI techniques for predictive insights, fraud detection, and optimization.
- **Intelligent Automation & NewTech**
Implements RPA, cognitive automation, and hyper automation to streamline processes and improve efficiency.
- **Digital Solution Architecture**
Designs scalable, cloud-based, and API-integrated technical architectures to support digital innovation.

Together, these teams enable end-to-end data solutions from strategy to implementation ensuring KPMG's clients are prepared for the future of business.

My internship was situated in the **Intelligent Automation & NewTech** team, where I contributed to two projects aimed at automating manual processes and enhancing decision-making through AI and large language models: focusing on two impactful automation tools: the Slide Deck Generator and the Banking Control Assessment system.



Figure 3 : The Different departments at KPMG Lighthouse

2. Internship Assignment

This section introduces the two intelligent automation projects developed during my internship at KPMG Lighthouse, the center of excellence for data, AI, and emerging technologies within KPMG Belgium. These initiatives aimed to solve real-world business challenges through scalable and maintainable AI-powered tools.

2.1. Slide Deck Generator

An AI-based tool designed to automate the creation of standardized, on-brand slide presentations using a Retrieval-Augmented Generation (RAG) pipeline. The solution streamlines content extraction and slide formatting, significantly reducing manual effort while ensuring consistency and quality.

2.2. Banking Control Assessment

A cognitive automation solution that leverages Large Language Models (LLMs) to analyze internal banking controls, detect duplicates, and assess quality and risk factors. The tool supports audit and compliance teams by improving the efficiency and accuracy of control evaluations.

3. Analysis

This chapter provides an overview of the tools and technologies used for the two major components of my internship project: **the Slide Deck Generator and the Banking Control Assessment tool**. Although both tools addressed distinct business challenges, they shared a common architectural core centered around Large Language Models (LLMs), vector databases, and orchestration frameworks.

The realization of these tools began with Streamlit to rapidly prototype the user interfaces and core functionalities. As complexity increased particularly in both projects the need for a modular, visual, and collaborative framework led to the adoption of Langflow. Throughout both phases, technology decisions were based on trade-offs between performance, flexibility, ease of use, and future maintainability.

3.1. Tools Used – Overview and Purpose

Below is a combined and elaborated list of the tools and libraries used across both the Slide Deck Generator and the Banking Control Assessment projects. Each tool was selected based on its compatibility with the tech stack, community support, and the specific requirements of the use cases.

Streamlit

Used to rapidly prototype and deploy interactive user interfaces. It allowed me to quickly visualize the functionality of both tools and gather feedback from my mentors and managers early in development.

Langchain

Served as the backbone for building complex chains of operations involving LLMs. It provided modular abstractions to combine tools, prompts, memory, and agents in a flexible way.

Langchain-OpenAI

This package enabled seamless integration between Langchain and OpenAI's models (like GPT-3.5 and GPT-4), which were used for tasks such as summarization, classification, and dynamic reasoning.

LangGraph

A Langchain extension that allowed for graph-based agent control flows. It was particularly useful for structuring non-linear workflows, such as decision trees or conditional prompts.

Langflow

A visual interface built on top of Langchain, used in the later stages of development especially for the banking control tool. It enabled easier visualization, debugging, and

modification of LLM pipelines through a node-based interface, which also made collaboration with non-developers easier.

FAISS (Facebook AI Similarity Search)

A high-performance vector store used to index and search document embeddings based on semantic similarity. It was crucial for implementing the retrieval step in our RAG (Retrieval-Augmented Generation) pipelines.

DuckDuckGo-Search

This tool was used to query external information from the web, which enriched the generated slides with fresh, relevant content. It enhanced the value of the Slide Deck Generator beyond static input sources.

PyPDF / PyMuPDF

Libraries used for parsing and extracting text from PDFs. This functionality was key to ingesting client documents and feeding their content into the generation pipeline.

Pydantic

A data validation and settings management library that ensured all input/output structures especially in Langchain tools were well-defined and type-safe. This helped avoid runtime issues and improved code reliability.

Python-dotenv

Used to securely manage API keys and environment variables, keeping sensitive credentials out of the main source code and enabling easier deployment across different environments.

Pandas

A powerful library for working with tabular data. It was especially helpful during the Banking Control Assessment phase, where financial control records and CSV-based inputs were loaded, transformed, and analyzed.

3.2. Vector Store Comparison: FAISS vs Alternatives

In the Slide Deck Generator project, retrieving semantically relevant content was essential. To support this, a vector store was required to store embeddings and perform fast similarity searches efficiently. After evaluating several options FAISS, Chroma, Pinecone, and Weaviate, I selected FAISS based on hands-on testing, research into official documentation, and community benchmarks. (FAISS, s.d.)



The tools were compared using key criteria: speed, ease of use, community support, and integration with Langchain.

Tool	Speed	Ease of use	Community	Integration
FAISS	4	3	4	4
Chroma	3	4	3	4
Pinecone	4	4	4	3
Weaviate	3	3	3	3

Table 1 - Weighted Decision Matrix for Vector Store Selection

3.3. Why FAISS Was Selected

FAISS was ultimately selected for the **Slide Deck Generator** due to its high performance, solid integration with Langchain, and most importantly its ability to run **fully offline**. Since the tool was developed and tested locally without relying on cloud infrastructure, FAISS provided the best balance of speed, simplicity, and privacy. Other tools like Pinecone and Weaviate were ruled out due to their reliance on external hosting and less favorable offline support. (TiDB, 2024)

3.4. Langflow vs Streamlit + Langchain

As the logic for both projects evolved, particularly for the Banking Control Assessment, it became clear that managing Langchain chains purely in code through Streamlit introduced limitations. While Streamlit enabled fast prototyping, its linear structure and lack of visualization made it harder to debug and scale complex workflows involving multiple tools, memory objects, and branching logic.

To address these challenges, Langflow was adopted in the later development stages. Langflow is a visual orchestration tool built on top of Langchain that allows users to design and connect components (tools, prompts, chains, etc.) in a node-based editor. This shift enabled better collaboration, improved clarity in chain structure, and faster iterations especially when gathering feedback from non-technical stakeholders.

The table below provides a comparison of the two approaches based on key criteria:

Criteria	Streamlit + Langchain (code-based)	Langflow (visual-based)
Development Speed	Medium (fast to start, slower later)	Fast (once setup is done)
Chain Visualization	None	Built-in
Collaboration	Developer only	Accessible to non-devs
Flexibility / Custom Logic	High (full code control)	Moderate (limited GUI nodes)
Reusability of Components	High	Moderate
Debugging Experience	Manual	Visual & interactive

Table 2 - Weighted Comparison of Workflow Tools

The combination of Streamlit and Langchain was ideal for initial experimentation and quick MVP creation. However, as the complexity of the agent workflows increased especially for dynamic banking control validation the move to **Langflow** significantly improved maintainability, reduced errors, and made the logic transparent to a broader team. This hybrid approach allowed for both flexibility and scalability during the project lifecycle.

4. Project Timeline

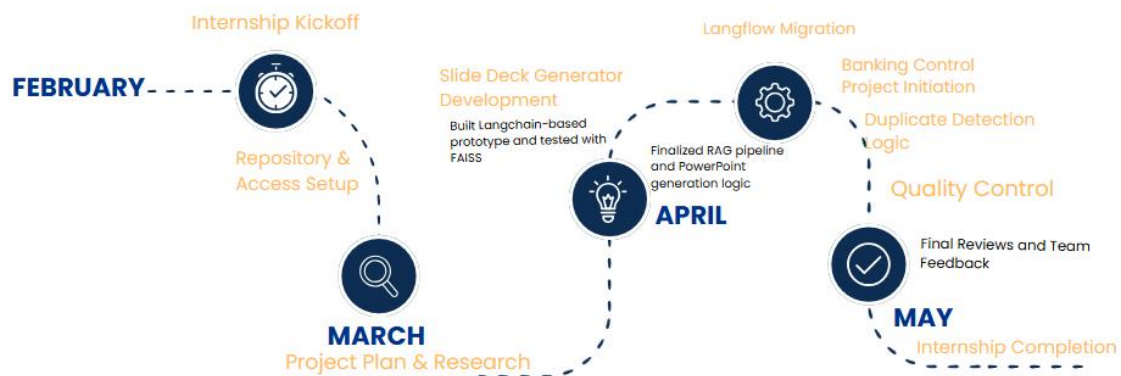


Figure 4 - Internship Project Timeline (2024–2025)

To ensure a structured and focused delivery, the internship was organized around a progressive monthly timeline. Each phase built upon the last from early onboarding and research, to development, testing, and final documentation. This roadmap helped align technical work with stakeholder expectations and ensured both projects progressed in parallel where possible.

5. Project 1 : Slide Deck Generator

The Slide Deck Generator is an internal AI tool built at KPMG Lighthouse by the Intelligent Automation and NewTech Team to automate the creation of standardized pitch presentations. It was designed to support consultants by significantly reducing the manual effort typically spent drafting, formatting, and structuring slides. Initially implemented using LangChain and FAISS, the solution was later migrated to a Langflow-based architecture for greater modularity and maintainability. It enables users to upload internal reports, retrieve key insights, and export a fully formatted .pptx file all within minutes.

5.1. Use Case and Motivation

Creating branded pitch decks can take up to 3 hours per project due to repetitive formatting and manual information extraction. While several AI-based tools like Gamma or Slidesgo exist, they often operate as external SaaS platforms and are not suitable for handling confidential client data.

The Slide Deck Generator was developed as an internal solution tailored to KPMG's standards and privacy requirements. It leverages AI to extract relevant insights from client documents and insert them into a predefined slide template. The goal is to save time, ensure visual and brand consistency, and support scalability across multiple consulting use cases all while ensuring that sensitive data remains securely managed within the organization's infrastructure.

The screenshot displays the Streamlit user interface for the KPMG Slide Deck Generator. At the top, a dark blue header bar contains the KPMG logo on the left and a 'Welcome!' message on the right. Below the header, the main title 'Generate your pitch deck (AI-Powered)' is prominently displayed. Underneath the title, there are two tabs: 'Create Slide Deck' (which is the active tab) and 'Advanced Configuration'. The 'Create Slide Deck' tab is divided into two main sections. The left section, titled 'Step 1: Enter the target audience', contains a 'Client Name' input field with a placeholder text 'e.g. Microsoft, Apple, Google'. Below this is an 'Upload PDF files' section, which includes a 'Drag and drop files here' area with a note 'Limit 200MB per file • PDF' and a 'Browse files' button. At the bottom of this section is a 'Generate Slides' button and a checkbox labeled 'Reset vector store before upload'. The right section, titled 'How it works', lists four steps: 1. Enter the name of Client, 2. Upload relevant documents, 3. Click 'Generate Slides', and 4. Review and download your presentation. In the top right corner of the interface, there is a 'Deploy' button.

Figure 5 : Streamlit UI with upload & generate interface

5.2. Tools and Technologies Used

A detailed explanation of the tool stack used throughout the internship is available in [Section 3.1](#). This section offers a concise summary specifically for the Slide Deck Generator, accompanied by Figure 6, which visually maps the core technologies.

The project began using LangChain to develop the Retrieval-Augmented Generation (RAG) pipeline. While this offered modularity and control, it was later replaced by Langflow, which provided a visual interface that improved maintainability, debugging, and collaboration, especially during iterations with mentors. To handle the language model operations and embedding generation, Azure OpenAI was used across all stages.

For the vector database, FAISS was initially chosen due to its offline capabilities and strong compatibility with LangChain. However, I later transitioned to PGVector, as it provided better integration with PostgreSQL, which was already used internally at the company. PGVector also worked more smoothly within Langflow's multi-flow environment, making it the more practical long-term choice.

User interaction was managed via Streamlit, which allowed rapid prototyping of the UI and enabled document upload, client input, and result previewing. Once the AI-generated content was ready, Azure Functions handled the automation of PowerPoint slide generation by inserting JSON content into a predefined template. Finally, PyMuPDF was used at the start of the pipeline to extract text from uploaded PDFs, feeding clean content into the embedding step.

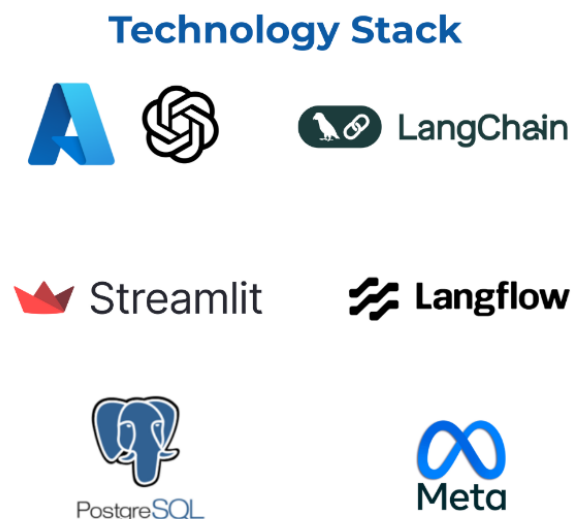


Figure 6 : Technology Stack Image

5.3. Architecture and Design

To simplify the development and allow easier visualization of the pipeline, we transitioned from a purely code-based setup to Langflow. Langflow allowed us to define and execute flows using a visual interface, improving maintainability, onboarding, and clarity.

The system is divided into two main flows:



Pitch Deck Data Loader Edited 12 days ago

Upload zip file, create and store embeddings (for all files in zip) in PG Vector store.



Pitch Deck Generator Edited 14 days ago

Create a pitch deck from template by querying PG vector store with Agent and passing results to Azure function via API.

A) PITCH DECK DATA LOADER :

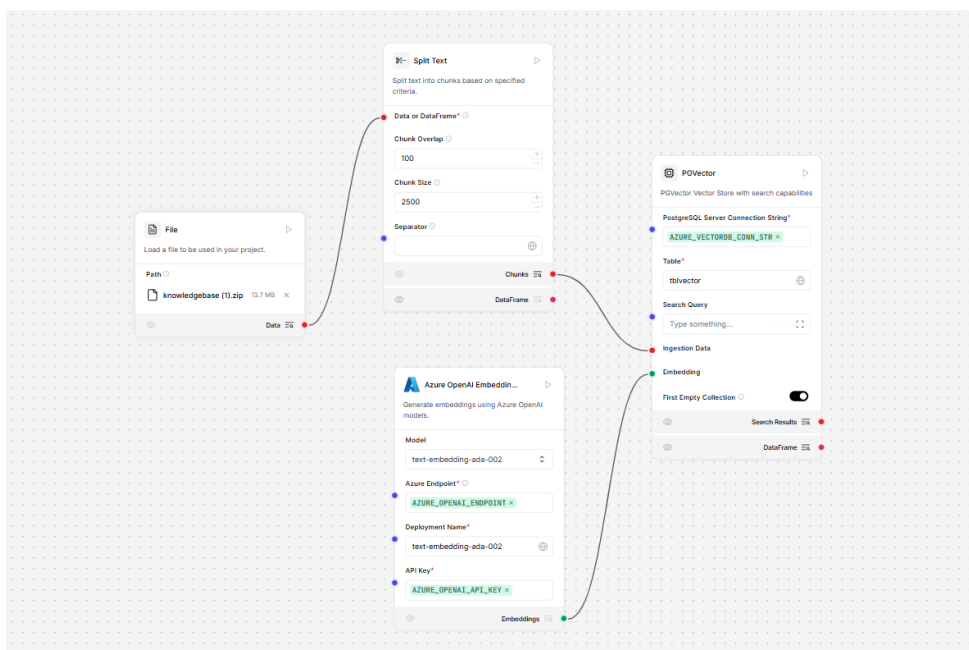
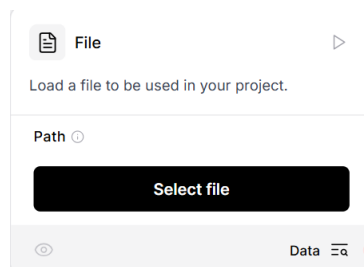


Figure 7 : Data Loader Langflow .Higher-resolution version of this figure is available in the Appendix (see Figure A1).

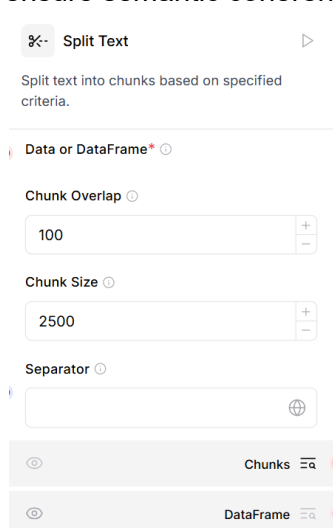
This flow is triggered when a user uploads one or more documents. The uploaded files are zipped and passed to the **File** node. The pipeline then proceeds as follows:

- **File Node** : Accepts the zipped user-uploaded PDF documents.



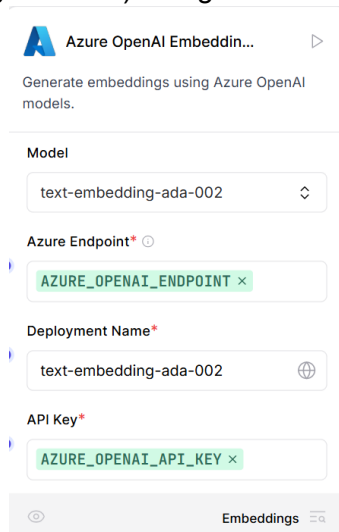
The screenshot shows the 'File' node interface. At the top, there is a document icon and the label 'File'. Below this, a text prompt says 'Load a file to be used in your project.' Underneath, there is a 'Path' label with a help icon. A large black button with the text 'Select file' is centered below the path label. At the bottom of the node, there are two tabs: 'Data' (selected) and 'DataFrame', each with a help icon.

- **Split Text**: Splits the full text into overlapping chunks (e.g., 1500 characters with 100-character overlap) to ensure semantic coherence.



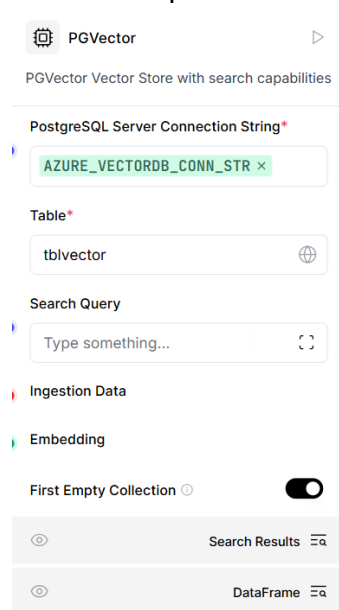
The screenshot shows the 'Split Text' node interface. At the top, there is a document icon and the label 'Split Text'. Below this, a text prompt says 'Split text into chunks based on specified criteria.' Underneath, there is a 'Data or DataFrame' label with a help icon. Below this, there are three input fields: 'Chunk Overlap' with a value of 100, 'Chunk Size' with a value of 2500, and 'Separator' which is empty. Each input field has a help icon and a '+'/- button. At the bottom of the node, there are two tabs: 'Chunks' (selected) and 'DataFrame', each with a help icon.

- **Azure OpenAI Embedder:** Sends chunks to the Azure OpenAI embedding model (e.g., text-embedding-ada-002) and generates vector representations.



The screenshot shows the configuration page for the 'Azure OpenAI Embedder'. At the top, there's a header with the Azure logo and the text 'Generate embeddings using Azure OpenAI models.' Below this, there are four input fields: 'Model' (set to 'text-embedding-ada-002'), 'Azure Endpoint*' (set to 'AZURE_OPENAI_ENDPOINT'), 'Deployment Name*' (set to 'text-embedding-ada-002'), and 'API Key*' (set to 'AZURE_OPENAI_API_KEY'). Each field has a small 'x' icon to clear the text. At the bottom, there's a 'Search Results' section with a 'DataFrame' tab selected.

- **PGVector:** Stores the resulting vectors in a PostgreSQL-backed vector database, enabling semantic search in later steps.



The screenshot shows the configuration page for 'PGVector'. It has a header with the PGVector logo and the text 'PGVector Vector Store with search capabilities'. Below this, there are three input fields: 'PostgreSQL Server Connection String*' (set to 'AZURE_VECTORDB_CONN_STR'), 'Table*' (set to 'tblvector'), and 'Search Query' (set to 'Type something...'). There are also sections for 'Ingestion Data' and 'Embedding'. At the bottom, there's a 'First Empty Collection' toggle switch and a 'Search Results' section with a 'DataFrame' tab selected.

This preprocessing step ensures that all input content is accessible and searchable via vector similarity, enabling better RAG performance downstream.

B) PITCH DECK GENERATOR :

The Pitch Deck Generator flow automates the creation of a personalized PowerPoint presentation based on uploaded documents and user input. After retrieving relevant content from a vector store, the system uses an AI agent to fill predefined placeholders (e.g., {title}, {fa_sales_23}) with context-specific data. The result is a fully structured JSON, which is then applied to a PowerPoint template, producing a ready-to-download deck. This flow ensures a smooth transition from raw input to professional output using Retrieval-Augmented Generation and template automation.

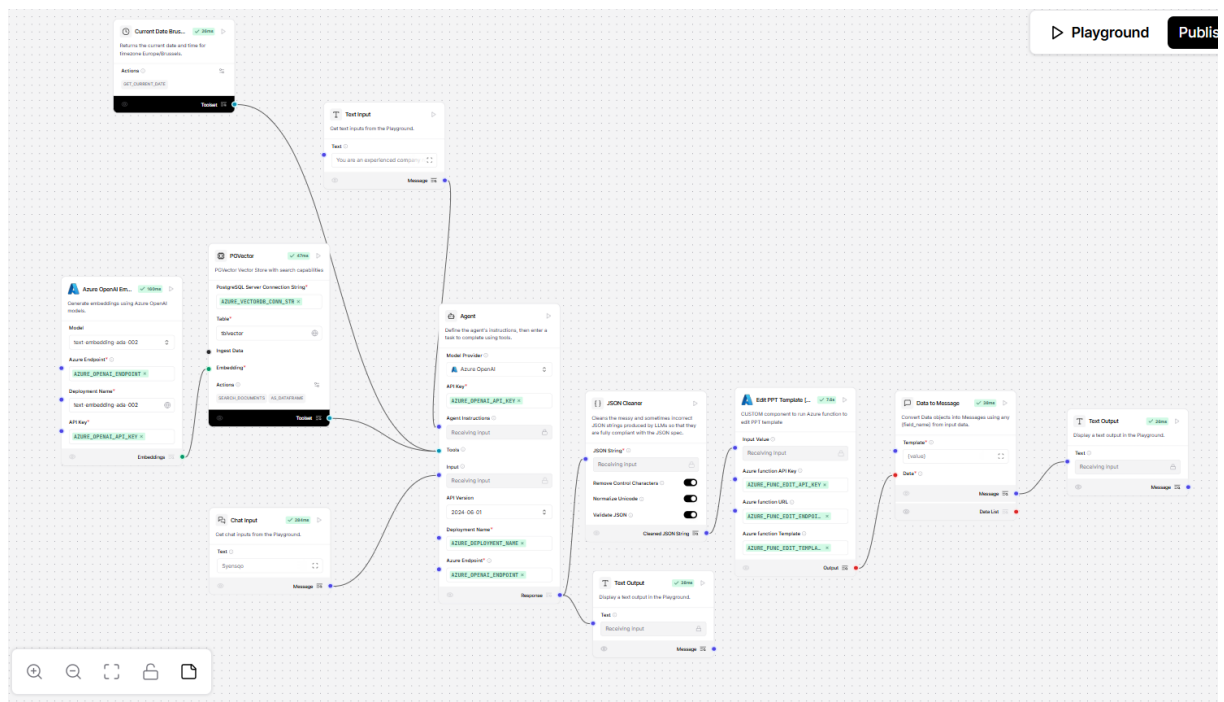


Figure 8 : Langflow: Slide Generator Flow Overview

- **Chat Input:** Accepts the user's input, typically the **client name** or context for the pitch deck. This value is passed into the agent as a dynamic variable used in both the query and the placeholder filling process.
- **Current Time Tool:** Fetches the current date and time, which can be used in the generated content (e.g., to populate {date} or contextualize insights as of today).
- **PGVector Tool :** Acts as the **retriever**. Given a query or placeholder name, it fetches the most relevant document chunks previously embedded and stored. These chunks are passed as context to the agent.
- **Agent**
The heart of the flow. It uses predefined **Agent Instructions** containing the target structure (placeholders like {title}, {fa_sales_23}, etc.). The agent performs RAG by combining the PGVector results and Chat Input, and generates a structured JSON output with all required values filled in.
- **JSON Output**
Captures the raw output from the agent, usually in JSON format with all the required keys filled.

- **JSON Cleaner**
Validates and cleans the JSON to ensure correct formatting, escaping of characters, and consistent structure essential for downstream automation.
- **Edit Template (Custom Component)**
A specialized node that takes the cleaned JSON and injects each key-value pair into the pre-configured PowerPoint template by replacing placeholders with values from the agent. For example, {title} in the slide becomes “Q2 Business Overview” if that's the agent's response.
- **Text Output**
Confirms success and makes the download link available for the newly generated .pptx presentation.

5.4. Implementation Highlights

The Slide Deck Generator combines AI reasoning, dynamic configuration, and template-based formatting to deliver a smooth, automated presentation generation experience. The following core mechanisms illustrate how the system transforms uploaded documents and user prompts into structured, branded PowerPoint slides:

- **Template-Based Generation**

The system uses a predefined PowerPoint template where each content block corresponds to a placeholder key (e.g., {title}, {fa_ratios}, {rm_key_risks}). These keys are filled automatically with AI-generated content during the flow execution.

- **RAG Logic (Retrieval-Augmented Generation)**

Langflow agents query a vector store (PGVector) to retrieve the most relevant content from the previously embedded documents. This allows the system to generate accurate, contextual slide content tailored to each client.

- **Advanced Prompting**

Prompts are carefully designed to enforce consistent formatting and data coverage. If no answer is found for a specific placeholder, the system gracefully defaults to "null" or an empty string to preserve template structure.

- **Configuration Editor**

A dedicated interface tab allows users to customize the list of placeholders and their associated instructions before starting the generation. This supports flexibility across business use cases while preserving structure.

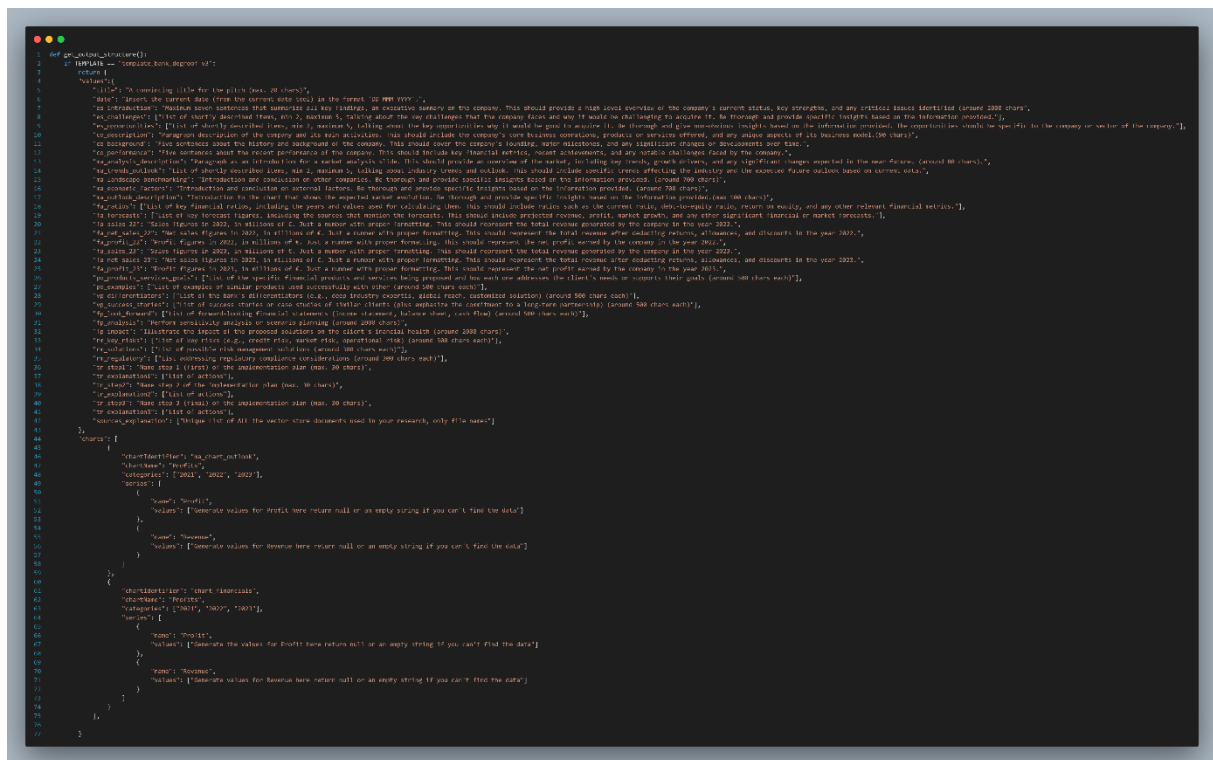


Figure 9 - Example of Structured Agent Instructions and Expected Output Format . Due to the length and density of this code snippet, a full-resolution version is included in Appendix A (see Figure A3) for easier readability.

Deploy

Edit the configuration values used to generate the pitch deck.

Reset to Default Values

	Parameter	Explanation	AI Instruction	Feedback
0	title	Title of your slide deck	A convincing title for the pitch	max
1	date	Month and Year of the presentation	Insert the current date (from the current date tool) in the format 'DD MMM YYYY'.	
2	es_introduction	Introduction to the client's business.	Maximum seven sentences that summarize all key findings, an executive summary or around 100 words.	around
3	es_challenges	Summary of key challenges.	List of shortly described items, min 2, maximum 5, talking about the key challenges to the business.	
4	es_opportunities	Potential opportunities	List of shortly described items, min 2, maximum 5, talking about the key opportunities for the business.	
5	co_description	Overview of the company's background, size, mission, etc.	Paragraph description of the company and its main activities. This should include the company's history, mission, and vision.	
6	co_background	Company background for the presentation.	Five sentences about the history and background of the company. This should cover the company's founding, growth, and current status.	
7	co_performance	Recent performance metrics and highlights.	Five sentences about the recent performance of the company. This should include key metrics, achievements, and challenges.	
8	ma_analysis_description	A short intro to the market analysis.	Paragraph as an introduction for a market analysis slide. This should provide an overview of the market and the company's position.	around
9	ma_trends_outlook	Summary of key industry trends and outlook relevant to the business.	List of shortly described items, min 2, maximum 5, talking about industry trends and the company's outlook.	

Figure 10 - Configuration Editor Interface for Slide Generation

5.5. Langflow–Streamlit Integration

To bridge user-friendly interactivity with backend AI automation, Langflow flows were integrated into a custom Streamlit interface using API-based communication. This allowed users to trigger Langflow pipelines through simple UI actions like file uploads or text inputs, while keeping logic and processing centralized in Langflow.

The integration consists of two main steps:

1. Uploading and Embedding Files

- Users upload PDF documents via Streamlit.
- The files are zipped and sent to the Pitch Deck Data Loader flow in Langflow.
- A specific API endpoint (/upload/{flow_id}) handles this upload.
- The uploaded documents are parsed using PyMuPDF, embedded using Azure OpenAI, and stored in PGVector for retrieval.

2. Generating the Presentation

- The user then enters a client name into the Streamlit interface.
- Streamlit triggers the Pitch Deck Generator flow by calling the /run/{flow_id} endpoint with structured inputs, outputs, and optional tweaks.
- Agent Instructions and Chart Instructions are dynamically injected through the tweaks parameter to tailor the system prompt for the Langflow agent.
- Once Langflow processes the request, the generated slide content (in JSON) is returned to Streamlit.
- The JSON is passed to a custom Azure Function that populates a PowerPoint template with the agent's output.

This integration design offers a clear separation of responsibilities:

- Langflow handles AI logic, vector retrieval, and prompt orchestration.
- Streamlit focuses on user interaction, input management, and output presentation.

5.6. Results and Outcomes

During internal testing, the generator successfully reduced the time required to produce a 10–15 slide pitch deck from what typically takes over 2 hours (Dbouk, 2019) to approximately 1-4min, depending on document length and complexity. While no formal user testing was conducted, this estimate is based on repeated manual benchmarking throughout development.

```
Total elapsed time pitch deck generation with RAG: 43.091299057006836
Total elapsed time pitch deck generation with RAG: 47.602909326553345
█
```

Figure 11 - Image showing the time it takes for slide generation

```
Total elapsed time pitch deck generation with RAG: 134.99057149887085
█
```

The generated slides consistently adhered to KPMG's brand guidelines and structure, improving clarity and reducing formatting errors. The modular design also opens the door for future enhancements, such as multilingual generation, dynamic slide review, and domain-specific templates.



[Demo Link](#)

6. Project 2 :Banking Control Assessment Automation

Operating with significant financial assets and sensitive data, financial institutions face substantial risks including fraud, regulatory breaches, and reporting errors. To mitigate these threats and ensure operational integrity, they rely on internal controls: systems of policies, procedures, and structured actions. These controls safeguard critical assets, enforce compliance, ensure data accuracy, and form the bedrock of a bank's stability and trustworthiness.

However, assessing whether these controls are properly designed and implemented is often done manually, making the process time-consuming, error-prone, and difficult to scale. To address this, KPMG Belgium developed a prototype that automates control assessment using large language models (LLMs) and semantic similarity techniques.

To tackle this challenge, KPMG Belgium developed a prototype that automates the evaluation of banking controls using AI technologies such as large language models (LLMs), vector similarity search, and structured prompt design. This project was carried out within the Intelligent Automation team and aimed to modernize the control assessment process with minimal user intervention while ensuring quality and traceability.

6.1. Use Case & Motivation

Organizations today face increasing regulatory complexity and the need for continuous assurance over their internal control environments. As both business operations and compliance obligations scale, the effort required to evaluate the design and effectiveness of internal controls grows proportionally often manually.

KPMG's **Banking Control Assessment** automation solution addresses this challenge by providing a tool that uses **Large Language Models (LLMs)** and **semantic similarity techniques** to assist in:

- Identifying duplicate or redundant controls
- Evaluating control quality using a scoring model
- Suggesting automation potential for specific controls
- Improving audit readiness and data consistency

The goal is to reduce manual effort, improve the consistency of assessments, and enhance insights into where automation and improvement opportunities lie.

6.2. Context and Business Value

This tool was developed as part of a wider initiative by KPMG's **Intelligent Automation** team to accelerate control assessments for internal audit, compliance, and GRC transformation projects.

Compared to traditional assessments that rely heavily on spreadsheets and interviews, this solution introduces:

- **Scalability:** Handles large volumes of controls across business lines
- **Data-driven automation:** Enables predictive scoring and semantic analysis
- **Audit quality:** Enforces structured frameworks like 5W1H and risk alignment
- **Time savings:** Reduces the time required for duplication checks and control reviews from hours to minutes

6.3. Tools & Technologies

The Banking Control Assessment solution combined previously introduced components with a few new technologies tailored to high-volume spreadsheet handling and backend logic. The system architecture was built using Langflow for visual orchestration, Streamlit for the frontend, and custom Python scripts for control logic and API integration. As discussed earlier ([see Section 3.1](#)), tools like OpenAI embeddings, PGVector, and Langflow handled the core AI workflow: embedding control definitions, querying similar entries, and generating structured outputs.

What was new in this project was the introduction of Polars and OpenPyXL, which replaced traditional Pandas-based processing. These tools significantly improved performance, especially when handling large Excel files containing complex control matrices. Polars enabled fast, memory-efficient transformations of control data, while OpenPyXL was used specifically to read Excel sheets with structured columns such as "Reference," "Risk," "Definition," and "Process." Together, they ensured smooth preprocessing of Risk Control Matrices (RCMs) prior to embedding.

Additionally, a set of custom Python utilities (`banking_funcs.py`, `worker.py`) handled API communication with Langflow, calculated cosine similarities between embedded vectors, and structured the JSON output for display. These utilities formed the functional backbone of the banking control logic, automating duplicate detection, quality scoring, and formatted result generation.

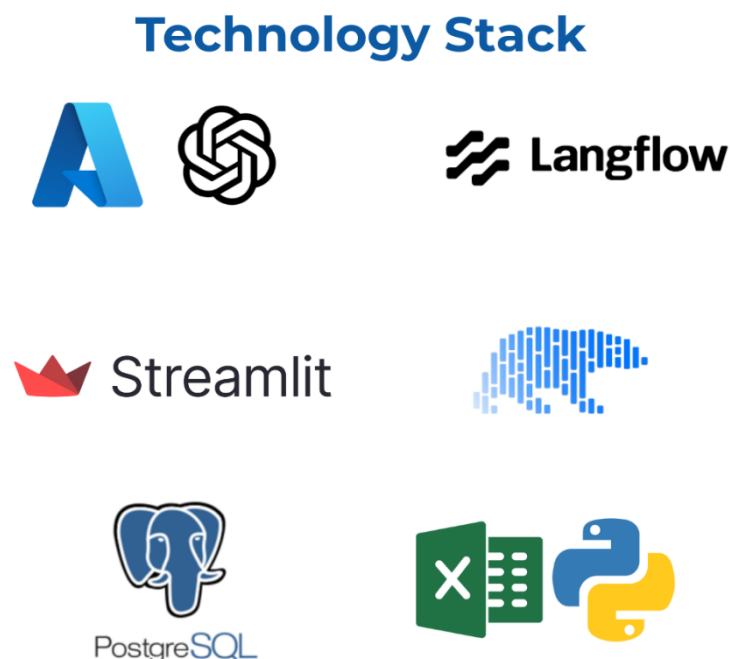




Figure 12 - Technologies used for Project 2 (Banking Control Assessment Automation)

6.4. Architecture & Implementation

The Banking Control Assessment system is designed to take an Excel file containing control definitions and return structured evaluations including quality scoring, duplication checks, and automation potential through an AI-enhanced workflow. The solution integrates **Langflow** for flow orchestration, **Streamlit** for the user interface, and several backend utilities for parsing, embedding, and similarity analysis.

For this, project , I worked on two main flows: the **Duplicate control** and the **Quality Control**

 **Quality Control - One agent** Edited 5 minutes ago
Conversational Cartography Unlocked.

 **Duplicate Control - Fab** Edited 12 days ago
Graph Your Way to Great Conversations.

A) DUPLIACATE CONTROL :

The Duplicate Control flow is designed to automatically detect semantically similar or redundant internal controls based on their descriptions. Manual duplication analysis is labor-intensive and often inconsistent, especially in large control libraries. This flow streamlines the process by embedding the control definitions, calculating similarity scores, and using an agent to analyze and explain the differences between potential duplicates.



Figure 13 - Duplicate Control Flow

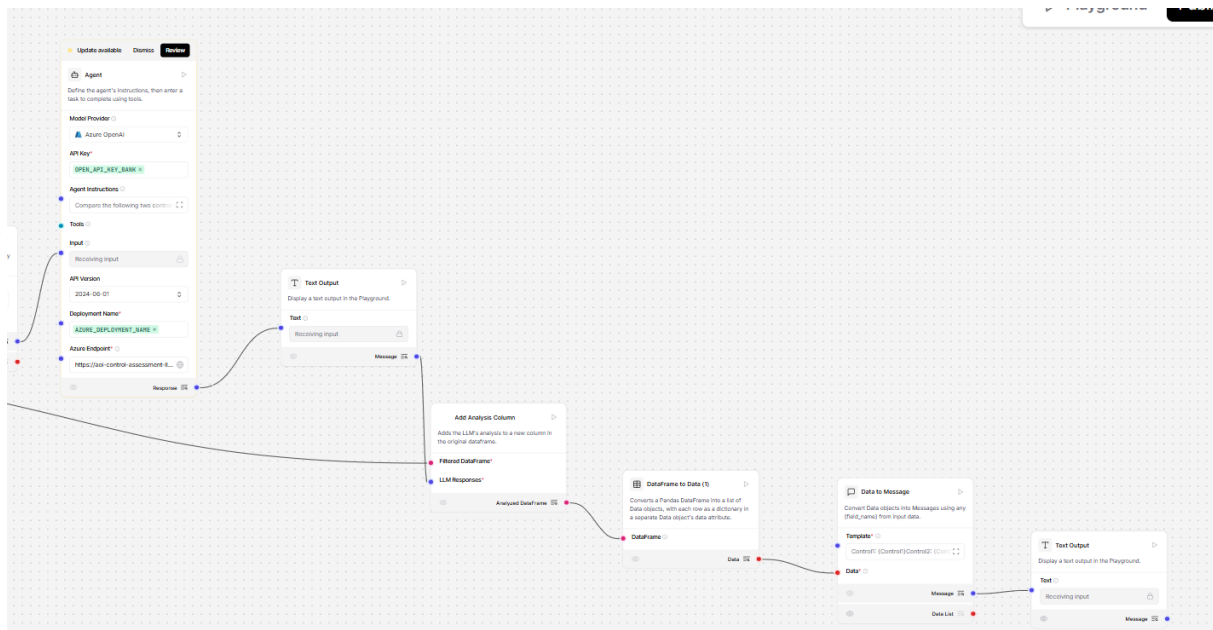


Figure 14 - Duplicate Control flow

WHAT THE FLOW DOES:

The duplicate control flow begins by accepting an Excel file containing internal control descriptions. These descriptions are first embedded using OpenAI embeddings, transforming them into numerical vectors that represent their semantic meaning. Once embedded, the system computes pairwise cosine similarity scores between the controls to identify potential duplicates. Pairs that exceed a predefined similarity threshold are retained for further analysis. Each of these high-similarity pairs is then passed to a Langflow agent, which compares the two controls and generates a qualitative explanation highlighting any redundancies or meaningful differences.

Component Output

Inspect the output of the component below.

Outputs

Logs

Control1	Control2	Similarity	Definition1	Definition2	Analysis
entity A.044	entity A.045	0.9956	1) Check that the entity A h	1) Check that the entity A h	Both definitions emphasize the requ

Figure 15 - Resulting DataFrame

KEY NODES EXPLAINED

The flow begins with the DataFrame Loader, which imports the Excel file containing control data into a format suitable for processing. The Azure OpenAI Embedder then converts the English control definitions into vector embeddings using the text-embedding-ada-002 model. These embeddings are appended to the dataset by the Embeddings Processor, preparing them for semantic comparison.

Next, the Basic Similarity Calculator computes cosine similarity scores between all control definitions, and the Similarity Threshold Filter retains only those pairs exceeding a specified threshold (e.g., 0.9) to minimize false positives. These filtered results are passed through a DataFrame to Data converter, which restructures the entries into a list format compatible with Langflow's agent interface.

Each pair is then transformed into a structured prompt by the Data to Message component. The Agent (Duplicate Analysis) receives these messages and generates a qualitative explanation, indicating whether the controls are redundant and, if so, how they differ. For debugging or testing purposes, the Text Output node can optionally display raw agent responses directly within Langflow.

Once processed, the Add Analysis Column step appends the agent's explanation to the dataset. Finally, the Final Data Export & Output node converts the enriched dataset into a polished format that can be shown in Streamlit or exported for reporting.

B) QUALITY CONTROL :

The Quality Control flow aims to assess the completeness and clarity of control descriptions based on the 5W1H evaluation framework: What, Why, How, When, Where, and Who. This evaluation supports audit readiness and helps identify controls that are poorly documented or unfit for automation.

This flow processes the uploaded Excel file, extracts the control text, and uses a Langflow agent to generate structured answers to quality questions. The responses are then scored and presented back to the user in a readable table.

WHAT THE FLOW DOES

This flow begins by accepting an Excel file containing internal control definitions. It extracts the control-level text and enriches it with OpenAI embeddings. The embedded content, along with an optional knowledge base, is then sent to a Langflow agent. This agent evaluates each control using a quality assessment framework based on the 5W1H method (What, How, When, Where, Who, Why, and Systems). Specifically, it answers questions such as: what the requirements are for a given control, how they should be evaluated, when and how frequently they should be checked, and which systems are involved. It also determines why the control matters, who is responsible for it, and where the control should be verified. Finally, the agent's responses are cleaned, converted into a DataFrame, and scored based on completion to support consistent analysis and prioritization.

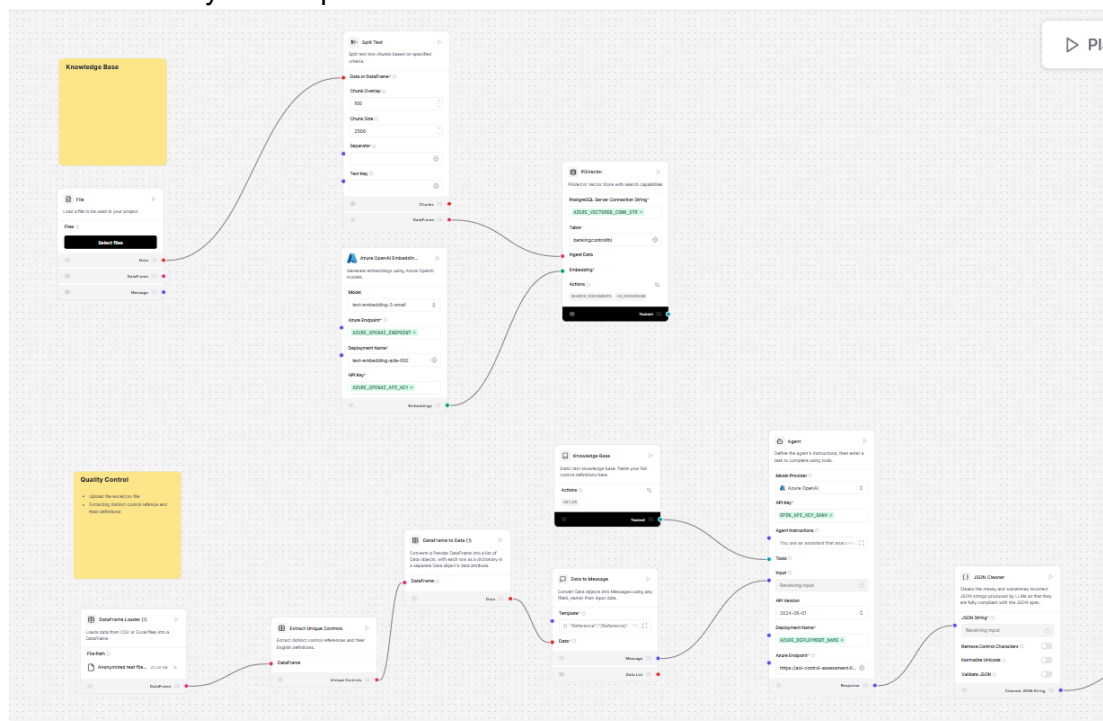


Figure 16 - Quality Control flow

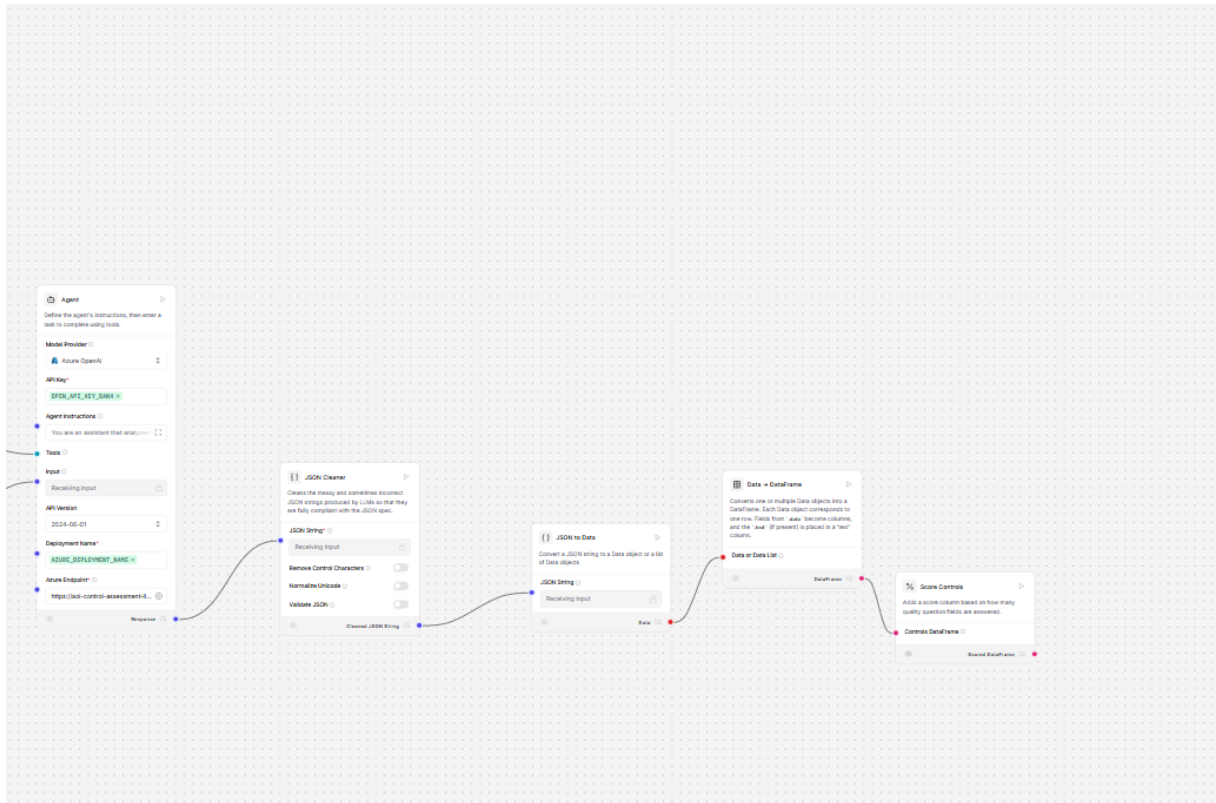


Figure 17 - Quality control flow

KEY NODES EXPLAINED

The flow begins with the **File + Split** Text component, which accepts the uploaded control document and breaks it into manageable chunks to ensure smooth processing during embedding. These chunks are then passed to the **Azure OpenAI Embedder**, which uses the text-embedding-3-small model to generate semantic vector representations of each control description. The resulting embeddings are stored in **PGVector**, enabling retrieval-augmented generation if needed by the downstream Langflow agent.

An optional **Knowledge Base** can also be connected, supplying supplemental context such as system definitions or glossary terms that help the agent deliver more accurate responses. To streamline input, the Extract Unique Controls node filters and deduplicates control rows, ensuring only distinct records are processed. These cleaned controls are converted into a list format by the **DataFrame to Data** node.

The **Data to Message** component takes each control block and structures it into a message template (e.g., “Reference: {{Reference}}, Definition: {{English definition}}”) so it can be interpreted by the agent. At the core of the flow is the Agent (Quality Assessor), which is configured with 5W1H-based instructions. It processes each control individually, generating structured responses to seven predefined quality dimensions.

Once the agent's output is received, it passes through a JSON Cleaner to remove escape characters and normalize formatting. The cleaned output is then converted back into a structured table using JSON to Data and Data to DataFrame components. Lastly, a custom Score Controls node evaluates the completeness of each agent response, assigning a score based on how many of the seven questions were fully answered.

Reference	English definition	What	How	When	Systems	Why
entity A.040	1. Ensure that the principles	The control requires docum	The requirements should be	This control should be chec		This
entity A.081	Based on a sample of entity	The criteria include the corr	The requirements should be	Checks should be conducte		This
entity A.195	Acquire Best Execution Rep	The control evaluates whetl	Requirements should be ev	This control should be chec		This
entity A.281	- Securities: Internal Contro	The control checks for com	Requirements should be ev	This control should be chec		This
entity A.029	[Make sure that all decision	The control ensures that as:	Evaluation should involve ri	This control should be chec		This
entity A.273	An ex-post control to check	The control checks for over	Requirements are evaluatec	This control should be chec	System A, System C, System	This
entity A.019	1. Ensure that entity A has c	The control ensures that ke	Evaluation should involve ri	This control should be chec		This
entity A.197	Acquire list of married trade	The control evaluates whetl	Requirements should be ev	This control should be chec		This

Figure 18 - Output of the quality control with the different questions answered and scored

Component Output

Outputs

Logs

Inspect the output of the component below.

	When	Systems	Why	Who	Where	Score ...
quirements should be	This control should be chec		This control is necessary to	Entity A is responsible for c		71
quirements should be	Checks should be conducte		This control is essential to	Responsibility lies with the		71
ements should be ev	This control should be chec		This control ensures that th	Entity A is responsible for c		71
ements should be ev	This control should be chec		This control is crucial to en	Internal Control is responsil		71
ition should involve ri	This control should be chec		This control is critical for co	Entity A is responsible for e		71
ements are evaluatec	This control should be chec	System A, System C, System	This control is essential to	Entity A and the Risk Depar		86
ition should involve ri	This control should be chec		This control is vital for mair	Entity A is responsible for e		71
ements should be ev	This control should be chec		This control is important to	Entity A is responsible for tl		71

6.5. Why Cosine Similarity?

In the Duplicate Control detection flow, **cosine similarity** was chosen as the primary method for comparing control definitions because it effectively captures **semantic similarity** between high-dimensional vectors generated from text embeddings.

When control descriptions are embedded using OpenAI's text-embedding-ada-002, each one is represented as a numerical vector in a high-dimensional space. Cosine similarity measures the **angle** between these vectors rather than their raw distance, which makes it ideal for determining **how similar two controls are in meaning**, regardless of their length or word count.

This approach offers several key advantages:

- **Scale-invariance:** Two controls with similar meaning but different word counts or phrasing still produce similar vectors.
- **Speed & Efficiency:** Cosine similarity is computationally lightweight and integrates seamlessly with PGVector and Langflow.
- **Fine-tuning:** A threshold (e.g., ≥ 0.90) can be applied to filter pairs with only strong semantic matches, reducing false positives.

By combining this similarity scoring with LLM-based reasoning, the system delivers both **quantitative** (similarity score) and **qualitative** (agent explanation) evidence for potential duplicates offering a comprehensive and explainable approach to control review.

6.6. Streamlit Integration & Final Outcome (Duplicate Control)

To enhance usability and allow business users to interact with the AI-powered control evaluation, the **Duplicate Control** flow was integrated into a custom **Streamlit** application. This integration allowed users to:

- **RCM File Upload via Streamlit:** Users can upload an RCM (Risk Control Matrix) Excel file directly within the Streamlit UI. This removes the need for backend interaction or manual file handling, allowing quick and secure ingestion of control data.

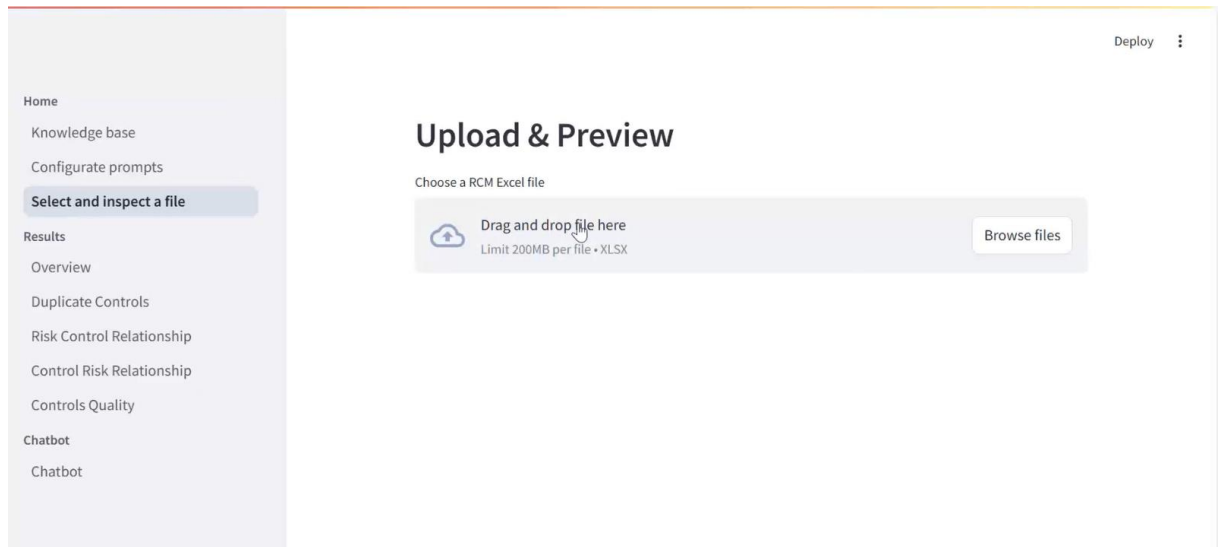


Figure 19 - Upload Tab of the Banking Control App

- **Langflow Flow Execution via API:** Once a file is uploaded, the user can trigger the Langflow pipeline with a single button. This is achieved via backend API calls that connect Streamlit to Langflow, initiating the duplicate detection flow and passing the uploaded file as input.

```

1  import httpx
2  import pandas as pd
3  from pathlib import Path
4  import json
5  import time
6  import sys
7
8  # Configuration
9  API_KEY = ""
10 BASE_API_URL = ""
11 FLOW_ID = ""
12 LOADER_NODE = ""
13 AnalysisNode = ""
14
15 def upload_file(flow_id: str, file_path: Path) -> str:
16     """Upload a file to LangFlow and return the file reference."""
17     print(f"Uploading file: {file_path}")
18     url = f"{BASE_API_URL}/api/v1/files/upload/{flow_id}"
19
20     with file_path.open("rb") as f:
21         try:
22             resp = httpx.post(
23                 url,
24                 headers={"x-api-key": API_KEY},
25                 files={"file": (file_path.name, f, "application/octet-stream")},
26                 timeout=60.0 # Increase timeout to 60 seconds
27             )
28             resp.raise_for_status()
29             file_ref = resp.json()["file_path"]
30             print(f"File uploaded successfully. Reference: {file_ref}")
31             return file_ref
32         except httpx.TimeoutException:
33             print("File upload timed out. The server took too long to respond.")
34             sys.exit(1)
35         except httpx.HTTPStatusError as e:
36             print(f"Upload failed with status {e.response.status_code}")
37             print(f"Response: {e.response.text}")
38             sys.exit(1)

```

Figure 20 - Langflow Integration to Streamlit

```

1 def run_flow(flow_id: str, file_ref: str):
2     """Run the flow with the file reference and return the raw output."""
3     print(f"Executing flow with file reference: {file_ref}")
4
5     # Note the URL format - this is often a source of 405 errors
6     url = f"{BASE_API_URL}/api/v1/run/{flow_id}?stream=false"
7
8     # Try the correct endpoint first - the one matching your error message
9     payload = {
10         "inputs": {
11             "LOADER_NODE": {"file_path": file_ref}
12         }
13     }
14
15     try:
16         # Attempt with increased timeout
17         resp = httpx.post(
18             url,
19             headers={
20                 "x-api-key": API_KEY,
21                 "Content-Type": "application/json",
22                 "Accept": "application/json",
23             },
24             json=payload,
25             timeout=120.0 # Increase timeout to 2 minutes
26         )
27         resp.raise_for_status()
28         return resp.json()
29     except httpx.TimeoutException:
30         print("Flow execution timed out. The operation may still be running on the server.")
31         print("Consider checking if the flow completed and retrieving results separately.")
32         sys.exit(1)
33     except httpx.HTTPStatusError as e:
34         print(f"Flow execution failed: {e.response.status_code}")
35         print(f"Response: {e.response.text}")

```

- **Real-Time Feedback on Duplicate Matches**

After flow execution, the number of detected duplicate controls is displayed immediately. This includes a summary count and optionally a preview of the matched control blocks, allowing users to quickly assess the output.

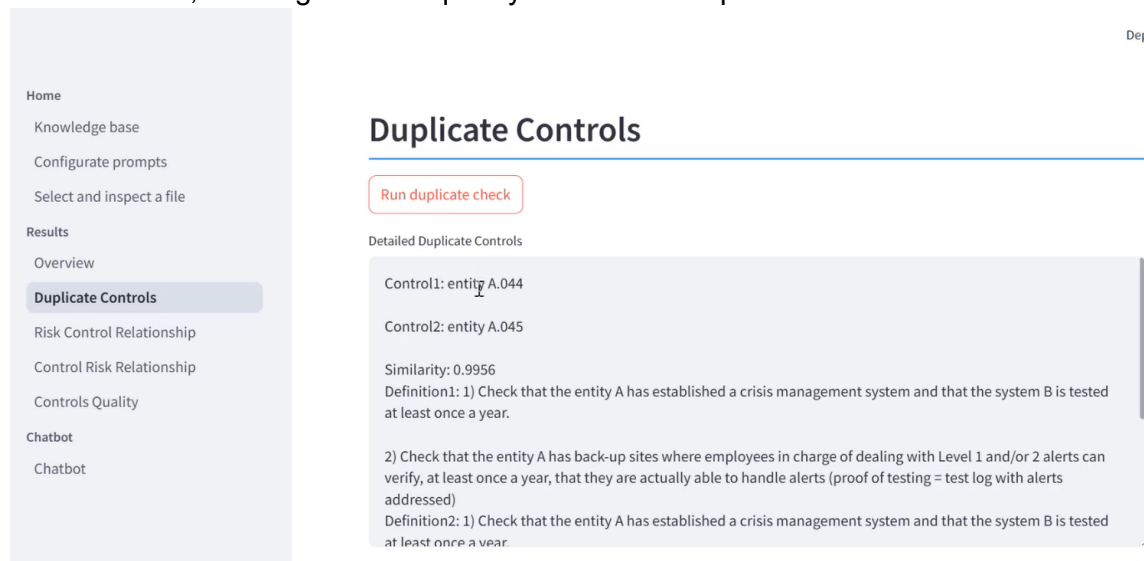


Figure 21 - Duplicate Tab with results

- **Agent-Based Explanation for Each Match**

For every control pair identified as potentially duplicated, the LLM agent provides a written explanation. This includes reasoning about whether the controls are redundant or similar, what differs, and whether merging or removal should be considered.

- **Result Export and Review**

The Streamlit app also allows users to export the results as a structured table. This enables further analysis, team review, or integration into broader audit documentation workflows.

How the Integration Works

- The Streamlit app first accepts and previews the uploaded control file.
- When the user clicks “**Run Duplicate Check**”, the file is sent to the Langflow API, which launches the Duplicate Control flow.
- The flow runs the embedding, similarity filtering, and LLM analysis as described earlier.
- Once complete, the output (agent response and similarity results) is returned and rendered as both:
 - A readable control block in a text area
 - A structured table showing matched control references and similarity explanations



[Demo Link](#)

7. Conclusion

Reflecting on this internship, I gained far more than just technical experience. I learned what it truly means to work in a large, structured organization like KPMG, one of the Big Four. It was an opportunity to contribute meaningfully while navigating the challenges of an enterprise-level environment.

Initially, I expected to work closely with a team, but the project turned out to be largely standalone. While that was unexpected, it pushed me to develop autonomy, self-discipline, and problem-solving skills. Over time, I came to appreciate the freedom to shape the project and the clarity that came from owning each part of the process.

Throughout the internship, I had the chance to build fully working AI tools, apply real-world prompt engineering, and deepen my understanding of LLM pipelines. Technologies like Streamlit and LangChain were familiar and gave me a strong foundation. However, Langflow was completely new. I had to self-learn, document extensively, and seek support from colleagues when necessary, especially in areas like agent configuration and data transformation.

Another takeaway was the mismatch between job descriptions and real-world scope. I was initially asked to familiarize myself with RPA tools like UiPath, but in practice, the project focused more on intelligent automation with LLMs. That pivot taught me to stay adaptable and focus on what's valuable, not just what's expected.

Beyond technical growth, this internship expanded my professional network and allowed me to meet inspiring people across departments. It reinforced the importance of being proactive, curious, and open to the unexpected.

Ultimately, I walk away with practical AI experience, a sharper understanding of enterprise workflows, and the confidence to build automation tools from scratch. For future interns, my message would be simple: don't expect a predefined roadmap, be ready to build your own.

8. Reference list

- [Meta AI. \(2017\). FAISS: A library for efficient similarity search. Retrieved from](#)
- Zilliz. (2023). *FAISS vs HNSWlib: Choosing the Right Tool for Vector Search*. Retrieved from <https://zilliz.com/blog/faiss-vs-hnswlib-choosing-the-right-tool-for-vector-search>
- LangChain. (2023). *Documentation*. Retrieved from <https://docs.langchain.com>
- Langflow. (2023). *Langflow Visual Orchestrator*. Retrieved from <https://docs.langflow.org>
- Streamlit. (2023). *Streamlit Docs*. Retrieved from <https://docs.streamlit.io>
- OpenAI. (2023). *Text Embedding Models*. Retrieved from <https://platform.openai.com/docs/guides/embeddings>
- PostgreSQL Global Development Group. (2023). *PostgreSQL Documentation*. Retrieved from <https://www.postgresql.org/docs/>
- Python Software Foundation. (2023). *OpenPyXL and Polars Libraries*. Retrieved from <https://openpyxl.readthedocs.io> and <https://pola.rs>
- Prompt Engineering. (2023, May 8). *Langflow: Drag and Drop ChatGPT Workflow Builder*. YouTube. <https://youtu.be/uA6sL65UNi4>
- AI Anytime. (2023, July 2). *LangChain vs Langflow*. YouTube. <https://www.youtube.com/watch?v=1ic-V0TCscM>

9. Bibliography

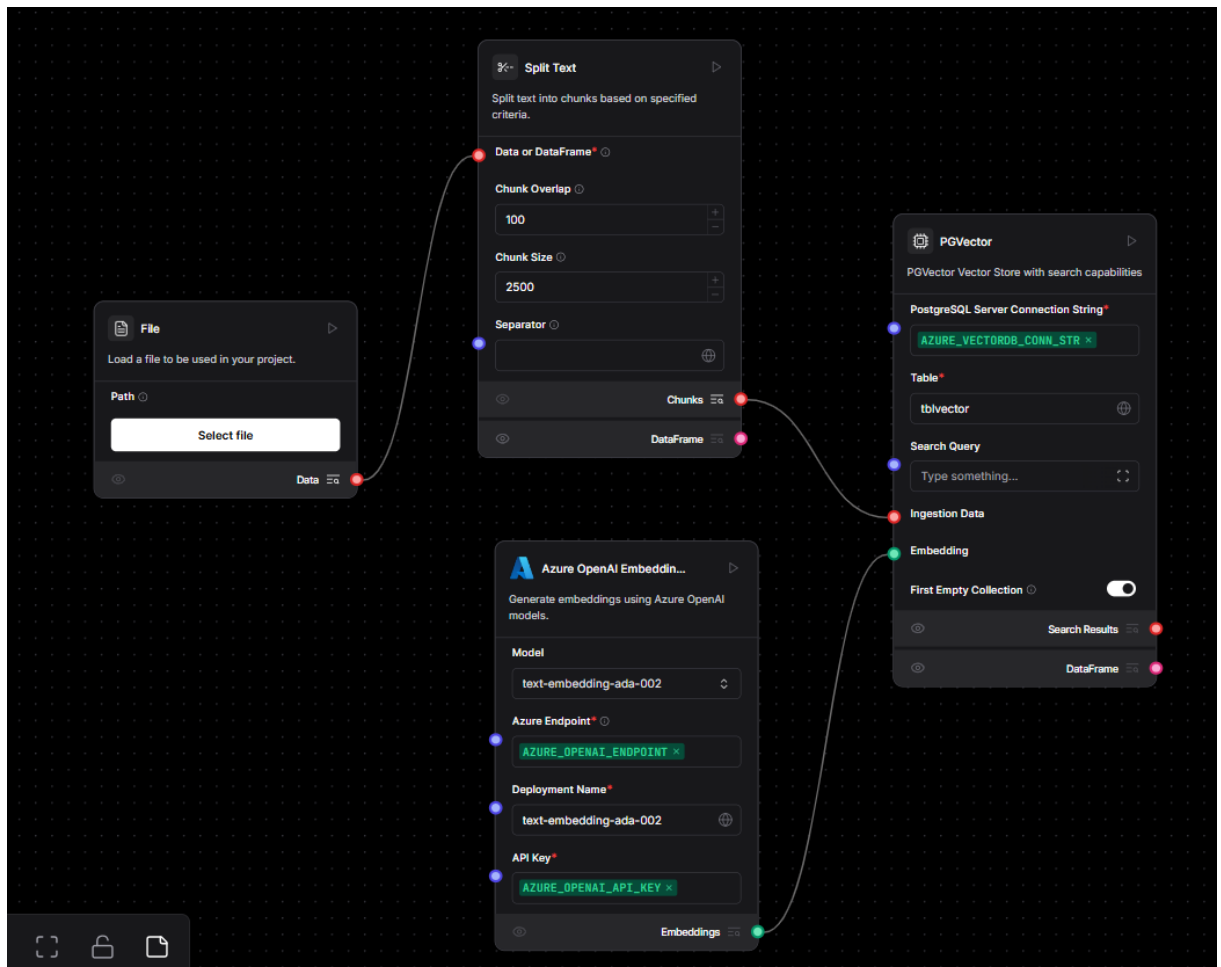
- Dbouk, J. (2019). *Quora*. Opgehaald van Quora: <https://www.quora.com/How-long-did-it-take-to-build-or-compose-a-pitch-deck-for-your-startup>
- FAISS. (sd). *Faiss Official Documentation*. Opgehaald van <https://faiss.ai/>
- TiDB, T. (2024, July 16). *TiDB*. Opgehaald van pingcap: <https://www.pingcap.com/article/mastering-faiss-vector-database-a-beginners-handbook/#:~:text=Scalability%3A%20Designed%20to%20handle%20datasets,suitable%20for%20real%2Dtime%20applications.>

10. Attachements

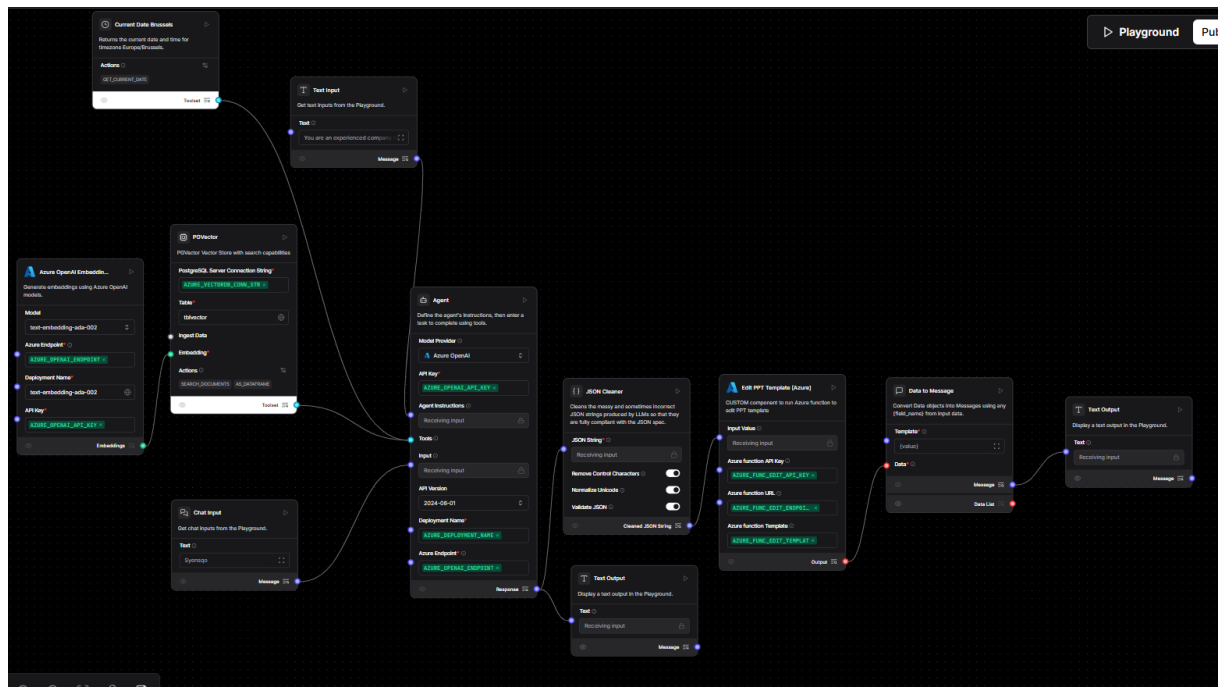
The following annexes provide additional context and technical detail that support the content presented in the main report. They include full-resolution diagrams, selected Langflow nodes, and code excerpts that were referenced but too dense to include inline. Each attachment has been referred to at least once in the body of the document.

APPENDIX A – VISUAL REFERENCES

A.1 Pitch Deck Data Loader



A.2 Pitch Deck Generator-fab-charts



A.3 Prompt Template and Output Structure for Slide Deck Generator

```
def get_output_structure():
    if TEMPLATE == "template_bank_degroof v3":
        return {
            "values":{
                "title": "A convincing title for the pitch (max. 20 chars)",
                "date": "Insert the current date (from the current date tool) in
the format 'DD MMM YYYY'.",
                "es_introduction": "Maximum seven sentences that summarize all key
findings, an executive summary on the company. This should provide a high-
level overview of the company's current status, key strengths, and any
critical issues identified (around 2000 chars",
                "es_challenges": ["List of shortly described items, min 2, maximum
5, talking about the key challenges that the company faces and why it would be
challenging to acquire it. Be thorough and provide specific insights based on
the information provided."],
                "es_opportunities": ["List of shortly described items, min 2,
maximum 5, talking about the key opportunities why it would be good to acquire
it. Be thorough and give non-obvious insights based on the information
provided. The opportunities should be specific to the company or sector of the
company."],
                "co_description": "Paragraph description of the company and its
main activities. This should include the company's core business operations,
products or services offered, and any unique aspects of its business model.(90
chars)",
                "co_background": "Five sentences about the history and background
of the company. This should cover the company's founding, major milestones,
and any significant changes or developments over time.",
                "co_performance": "Five sentences about the recent performance of
the company. This should include key financial metrics, recent achievements,
and any notable challenges faced by the company.",
                "ma_analysis_description": "Paragraph as an introduction for a
market analysis slide. This should provide an overview of the market,
including key trends, growth drivers, and any significant changes expected in
the near future. (around 80 chars).",
                "ma_trends_outlook": "List of shortly described items, min 2,
maximum 5, talking about industry trends and outlook. This should include
specific trends affecting the industry and the expected future outlook based
on current data.",
                "ma_landscape_benchmarking": "Introduction and conclusion on other
companies. Be thorough and provide specific insights based on the information
provided. (around 700 chars)",
```

```

    "ma_economic_factors": "Introduction and conclusion on external factors. Be thorough and provide specific insights based on the information provided. (around 700 chars)",
    "ma_outlook_description": "Introduction to the chart that shows the expected market evolution. Be thorough and provide specific insights based on the information provided.(max 100 chars)",
    "fa_ratios": ["List of key financial ratios, including the years and values used for calculating them. This should include ratios such as the current ratio, debt-to-equity ratio, return on equity, and any other relevant financial metrics."],
    "fa_forecasts": ["List of key forecast figures, including the sources that mention the forecasts. This should include projected revenue, profit, market growth, and any other significant financial or market forecasts."],
    "fa_sales_22": "Sales figures in 2022, in millions of €. Just a number with proper formatting. This should represent the total revenue generated by the company in the year 2022.",
    "fa_net_sales_22": "Net sales figures in 2022, in millions of €. Just a number with proper formatting. This should represent the total revenue after deducting returns, allowances, and discounts in the year 2022.",
    "fa_profit_22": "Profit figures in 2022, in millions of €. Just a number with proper formatting. This should represent the net profit earned by the company in the year 2022.",
    "fa_sales_23": "Sales figures in 2023, in millions of €. Just a number with proper formatting. This should represent the total revenue generated by the company in the year 2023.",
    "fa_net_sales_23": "Net sales figures in 2023, in millions of €. Just a number with proper formatting. This should represent the total revenue after deducting returns, allowances, and discounts in the year 2023.",
    "fa_profit_23": "Profit figures in 2023, in millions of €. Just a number with proper formatting. This should represent the net profit earned by the company in the year 2023.",
    "po_products_services_goals": ["List of the specific financial products and services being proposed and how each one addresses the client's needs or supports their goals (around 500 chars each)"],
    "po_examples": ["List of examples of similar products used successfully with other (around 500 chars each)"],
    "vp_differentiators": ["List of the bank's differentiators (e.g., deep industry expertis, global reach, customized solution) (around 500 chars each)"],
    "vp_success_stories": ["List of success stories or case studies of similar clients (plus emphasize the commitment to a long-term partnership) (around 500 chars each)"],
    "fp_look_forward": ["List of forward-looking financial statements (income statement, balance sheet, cash flow) (around 500 chars each)"],
    "fp_analysis": "Perform sensitivity analysis or scenario planning (around 2000 chars)",

```

```

        "fp_impact": "Illustrate the impact of the proposed solutions on
the client's inancial health (around 2000 chars)",
        "rm_key_risks": ["List of key risks (e.g., credit risk, market
risk, operational risk) (around 300 chars each)"],
        "rm_solutions": ["List of possible risk management solutions
(around 300 chars each)"],
        "rm_regulatory": ["List addressing regulatory compliance
considerations (around 300 chars each)"],
        "tr_step1": "Name step 1 (first) of the implementation plan (max.
30 chars)",
        "tr_explanation1": ["List of actions"],
        "tr_step2": "Name step 2 of the implementation plan (max. 30
chars)",
        "tr_explanation2": ["List of actions"],
        "tr_step3": "Name step 3 (final) of the implementation plan (max.
30 chars)",
        "tr_explanation3": ["List of actions"],
        "sources_explanation": ["Unique List of ALL the vector store
documents used in your research, only file names"]
    },
    "charts": [
        {
            "chartIdentifier": "ma_chart_outlook",
            "chartName": "Profits",
            "categories": ["2021", "2022", "2023"],
            "series": [
                {
                    "name": "Profit",
                    "values": ["Generate values for Profit here return
null or an empty string if you can't find the data"]
                },
                {
                    "name": "Revenue",
                    "values": ["Generate values for Revenue here
return null or an empty string if you can't find the data"]
                }
            ]
        },
        {
            "chartIdentifier": "chart_financials",
            "chartName": "Profits",
            "categories": ["2021", "2022", "2023"],
            "series": [
                {
                    "name": "Profit",
                    "values": ["Generate the values for Profit here
return null or an empty string if you can't find the data"]
                }
            ]
        }
    ]
}

```

```

    },
    {
      "name": "Revenue",
      "values": ["Generate values for Revenue here
return null or an empty string if you can't find the data"]
    }
  ]
},
],
}
}

```

A.4Langflow: Duplicate Control Detection Pipeline (Full Resolution)





A.5Langflow: Quality Assessment Flow (Full Resolution)

