NLP Project Report

Submitted by:

Syed Saad Waqar
(CT-22097)

Aarish Aijaz
(CT-22094)

Under Supervision of:

Miss Dure Shahwar

Department of Computer Science and Information Technology
NED University of Engineering & Technology, Karachi

Content

RAG-Based n8n Chatbot for Educational Institutions

Introduction

This project focuses on developing an intelligent data-driven system capable of responding to user queries using both relational databases and vector-based semantic search. By integrating PostgreSQL for structured data, Supabase for vector storage, and Python-based NLP algorithms, the system provides accurate, context-aware responses. The solution demonstrates modern AI techniques such as embeddings, tokenization, stemming, and summarization to extract meaningful insights from stored documents. It also supports hybrid retrieval, enabling both SQL-based factual answers and vector-based contextual answers. The system was built on **n8n**, a workflow automation platform, to orchestrate data flow and integrate the different components efficiently.

Overview

The system is designed as a hybrid information-retrieval chatbot that can handle diverse user queries. Structured data queries are processed through PostgreSQL, while unstructured content such as files, documents, and text data are stored as embeddings in a Supabase vector database. When the user provides a prompt, the system determines whether the query corresponds to SQL operations. The NLP module enhances context understanding by applying linguistic preprocessing and summarization. Using **n8n**, the project automates workflow execution, ensuring seamless integration between the database, vector store, and NLP modules. This combination ensures more accurate, relevant, and contextually aware responses.

Background and Motivation

Many applications today rely on both structured and unstructured data. Traditional SQL databases provide strong support for structured information but lack the ability to understand semantics or context. Meanwhile, organizations increasingly store text-based knowledge—policies, documents, notes, and reports—that cannot be effectively queried using SQL alone.

The motivation behind this system is to bridge this gap by combining relational querying with vector-based semantic search. Additionally, NLP techniques improve understanding of user intent and help extract core meaning from documents. The project aims to provide

a unified platform for answering queries intelligently and efficiently while reducing the manual effort required for information retrieval.

Methodology

The system follows a multi-step processing pipeline:

Data Preparation

- Text files and documents are uploaded to Google Drive and converted into embeddings.
- SQL datasets are stored in PostgreSQL tables.

NLP Preprocessing

- Tokenization, stemming, stopword removal, and frequency analysis are performed.
- Top words and summary extraction help determine document meaning and improve vector search relevance.

Vector Storage

- Processed embeddings are stored in Supabase's vector database.
- Metadata is linked for faster retrieval.

Query Handling

- User input is analyzed to determine whether it is SQL-related or context-related.
- SQL queries are executed on PostgreSQL.
- Contextual queries retrieve semantic matches from the vector database.

Response Synthesis

- Retrieved results and NLP summaries are combined to form a final, coherent response.

- **n8n** orchestrates the workflow and manages communication between the components.

Tool Description

The project uses the following tools and technologies:

- **n8n**: Workflow automation platform used to integrate PostgreSQL, Supabase, NLP modules, and APIs (n8n Workflows, 2025).
- **PostgreSQL**: For executing SQL queries and storing structured datasets (PostgreSQL Documentation, 2025).
- **Supabase Vector Database**: Stores text embeddings for semantic search (Supabase Docs, 2025).
- **Python**: Core programming language used for NLP preprocessing and query integration (Python Official Docs, 2025).
- **NLP Concepts**: Tokenization, stemming, frequency analysis, and summarization implemented in Python (Bird et al., 2009).
- **Embedding Models**: Converts text documents into numerical vectors for similarity matching (OpenAI Embeddings, 2023).
- **Gemini API**: For advanced AI model integration (Google Gemini API Documentation, 2025).
- **Google Drive API**: For accessing documents stored in Google Drive (Google Drive API Docs, 2025).

User Interface

The user interface provides a simple chat-based interaction system. Users type a question, and the backend determines whether to pull data from PostgreSQL or from vector embeddings. The interface displays results cleanly along with context extracted from documents. It ensures that even non-technical users can interact with the system without requiring knowledge of SQL or machine learning concepts.

Features

- **Hybrid Query Handling**: Automatically detects whether a query requires SQL lookup or semantic retrieval.
- **Vector-Based Search**: Retrieves the most relevant documents using cosine similarity.
- **NLP-Enhanced Understanding**: Uses tokenization, stemming, frequency analysis, and summarization.
- **Document Storage**: Files and textual data stored in Supabase are searchable through embeddings.
- **Secure Retrieval**: Uses authenticated connections for database communication.
- **Scalable Architecture**: Supports future expansion into more complex LLM reasoning.

Specification

- **Backend**: Python 3
- **Databases**: PostgreSQL, Supabase Vector DB
- **Libraries**: psycopg2 / SQLAlchemy, numpy, nltk (pure Python approach possible)
- **Environment**: Local development with cloud-based Supabase storage
- **Memory & Compute**: Embedding generation requires moderate CPU; vector queries are optimized on Supabase side.

Modularity of Analysis and Visualization

Overview

The system follows a modular structure where NLP analysis, vector operations, and SQL querying are handled in independent components. Each module can be updated or replaced without affecting the overall performance. Visualization and output formatting are separated from backend logic to maintain clean architecture.

Analysis

Analytical processing includes:

- Tokenization of input documents
- Stemming to normalize words
- Term Frequency (TF) analysis to identify important keywords
- Summary extraction to capture document meaning
- Vector similarity computation using cosine similarity to find relevant results

These processes allow the system to categorize text, extract themes, and interpret user intent accurately.

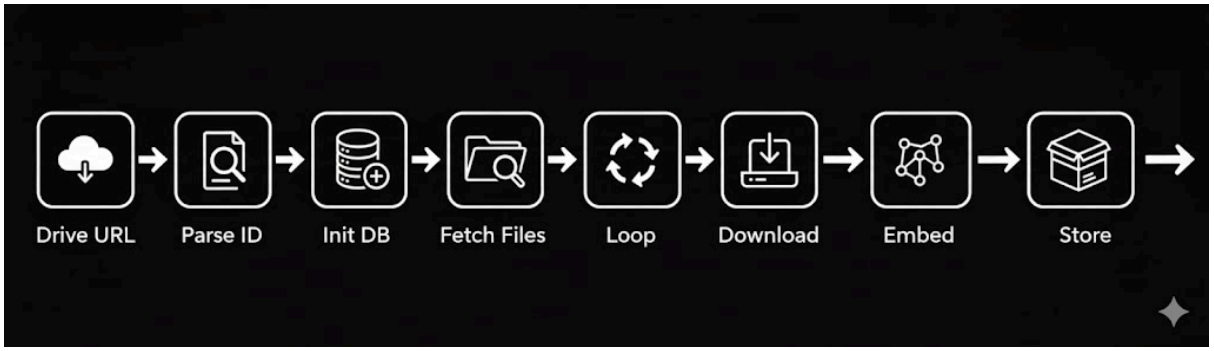
Visualization

The output is presented in a clean, readable format showing:

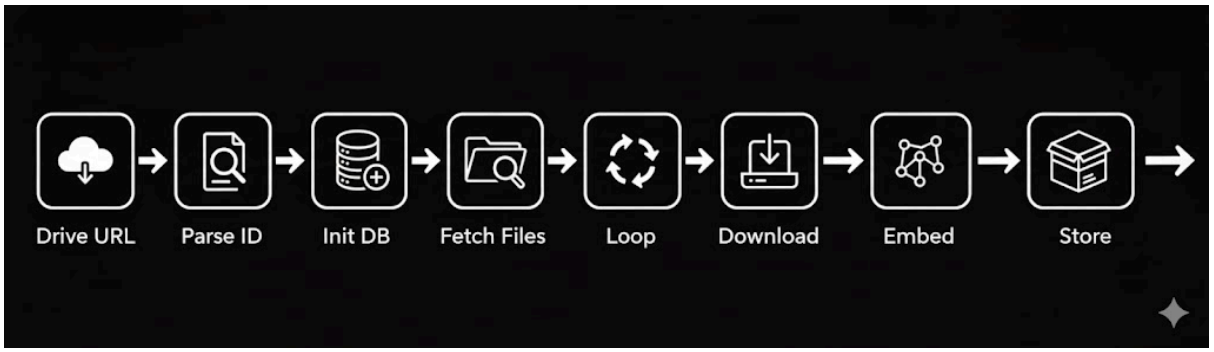
- Relevant SQL query results displayed as tables or structured text
- Semantic search results ranked by similarity score
- Optional highlighted keywords or summaries from NLP processing

The user interface ensures that retrieved knowledge is understandable and actionable.

Workflow 1:

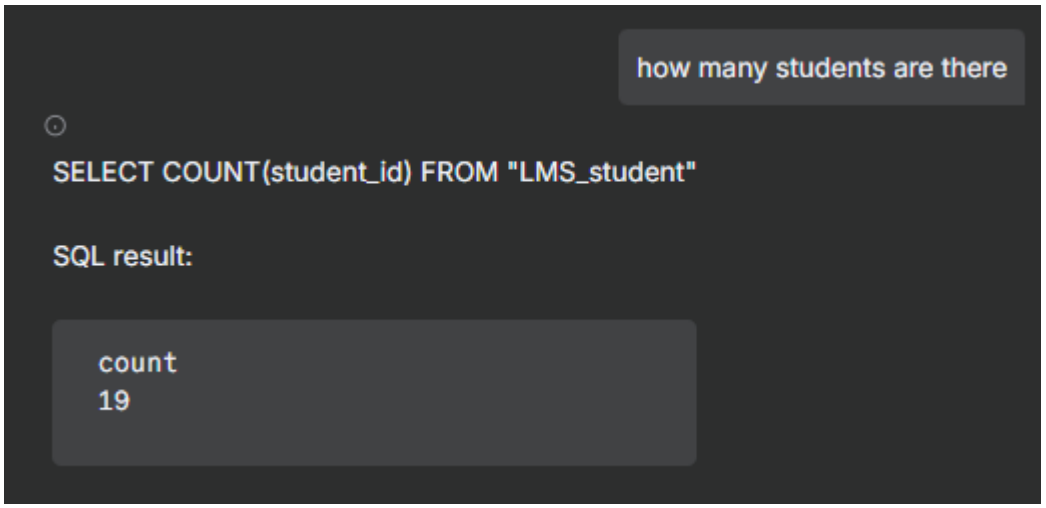


Workflow 2:

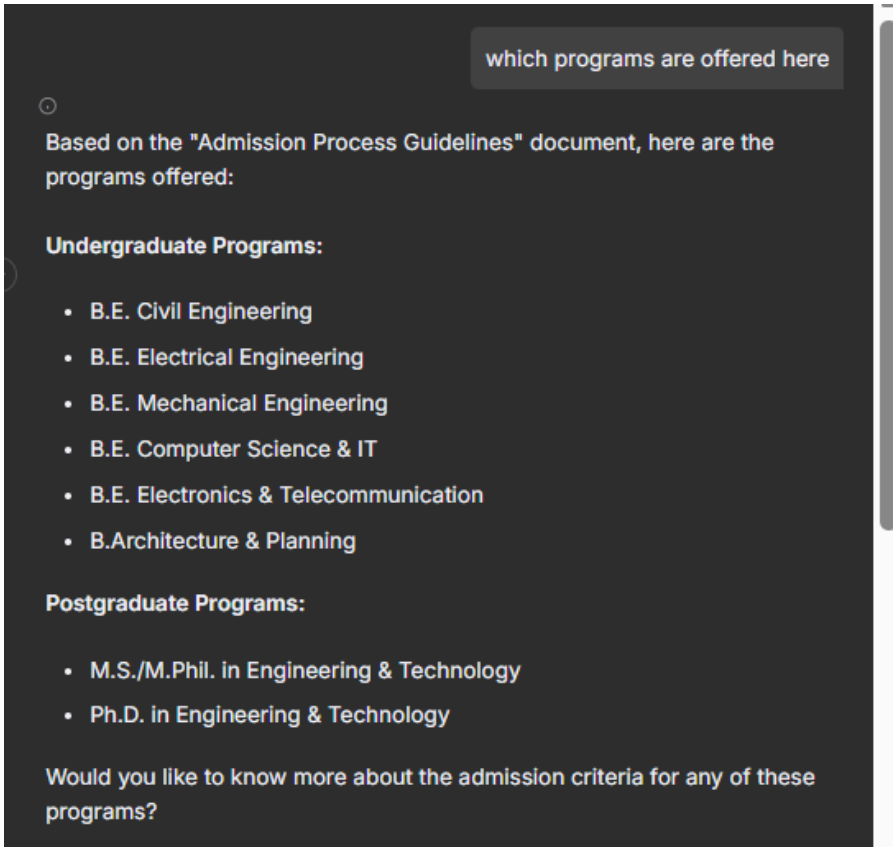


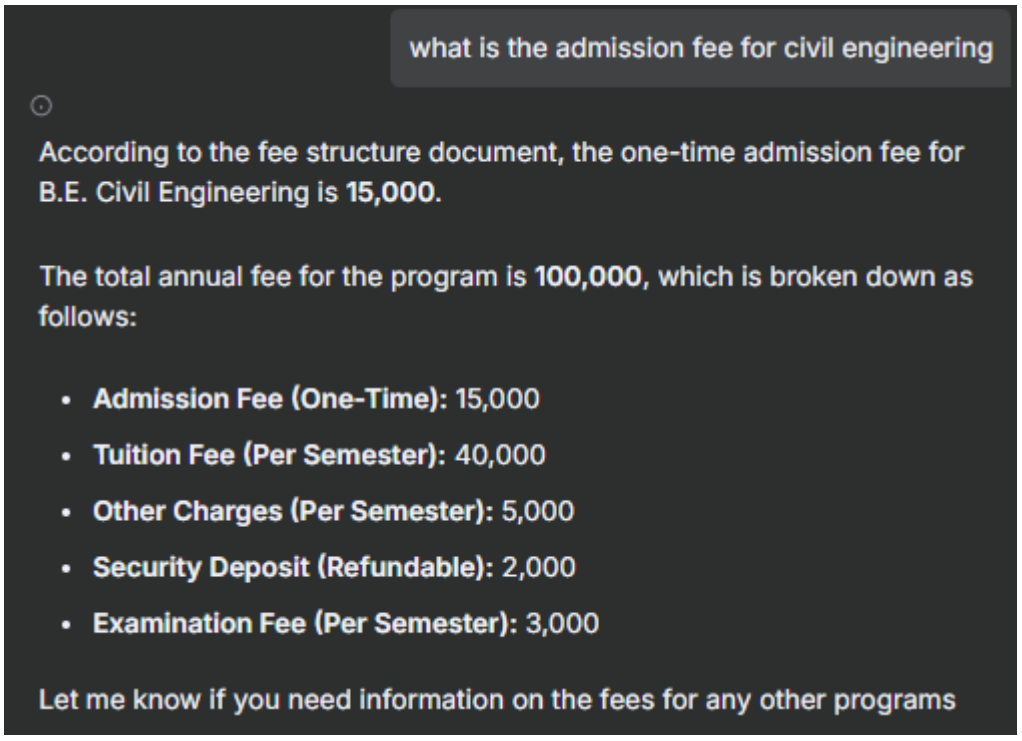
Sample Outputs:

Finds out how many students are there by running a SQL Query:



From the Documents uploaded in Drive it makes vector embeddings on Supabase it answers questions from vector embeddings in supabase:





Implementation

Implementation consists of the following steps:

1. Setting up PostgreSQL database and designing table schema.
2. Integrating Supabase and configuring vector storage.
3. Implementing Python scripts for:
    - Document ingestion
    - Embedding generation
    - SQL execution
    - NLP preprocessing
4. Building the core logic that routes queries to SQL or vector search.
5. Creating the chat interface for user interaction.
6. Testing with multiple document types and datasets.

Result and Discussion

The system successfully retrieves accurate results for both SQL queries and semantic searches. Users can interact with structured databases and unstructured documents through a single interface. Testing shows that vector search combined with NLP preprocessing greatly improves the relevance of contextual answers. The system demonstrates effective integration between relational data, embeddings, and NLP.

Future Work

- Integration of Fine Tuned Large Language Models (LLMs) for advanced reasoning

- Adding more file types (Excel, images with OCR)
- Improving query classification using machine learning
- Building a web dashboard for administrative control
- Supporting multi-language document embeddings
- Optimizing performance for large-scale datasets

References

1. PostgreSQL Documentation. (2025). *PostgreSQL official docs*. Retrieved from https://www.postgresql.org/docs/

2. Supabase Docs. (2025). *Vector Database with pgvector*. Retrieved from https://supabase.com/docs/guides/database/extensions/pgvector

3. Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media. Retrieved from https://www.nltk.org/book/

4. OpenAI. (2023). *Text and code embeddings guide*. Retrieved from https://platform.openai.com/docs/guides/embeddings

5. Google Gemini API Documentation. (2025). Retrieved from https://developers.google.com/gemini

6. Google Drive API Documentation. (2025). Retrieved from https://developers.google.com/drive/api/guides/about-sdk

7. Supabase Modules. (2025). *Vector storage and retrieval*. Retrieved from https://supabase.com/modules/vector

8. n8n Workflows. (2025). *Hybrid RAG Chatbots with Supabase and AI*. Retrieved from https://n8n.io/workflows/7381-create-a-knowledge-powered-chatbot-with-claude-supabase-and-