# RAG-Based LLM Error Resolution Bot for Windows11

## Introduction

The **RAG-Based LLM Error Resolution Bot for Windows 11** is an AI-powered troubleshooting tool designed to diagnose and resolve common Windows 11 errors. By leveraging Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs), this platform efficiently retrieves and processes error logs, troubleshooting guides, and FAQs to provide users with precise solutions.

Users can input error codes, keywords, or a brief description of their issue to receive natural language responses containing step-by-step troubleshooting instructions, drawn from a comprehensive knowledge base of official Windows documentation and community forums.

## Technologies Used

### 1. Streamlit: For building the user-friendly interface

Streamlit is a powerful, open-source framework for creating highly interactive and responsive web applications, particularly useful for data science and machine learning projects. It simplifies the process of building a user-friendly interface, allowing you to focus on functionality rather than complex front-end coding. In this context, Streamlit is used to create a seamless interface for users to interact with the system, such as uploading documents, inputting queries, or viewing generated solutions in a visually appealing way.

### 2. LLM (e.g., GPT-4): For generating natural language solutions

Large Language Models (LLMs) like GPT-4 are state-of-the-art in generating coherent and contextually relevant natural language responses. In this system, the LLM is leveraged to:

- Understand complex user queries.

- Analyze the extracted text.
- Generate accurate, human-like solutions, explanations, or summaries based on the user's needs. Its capability to process vast amounts of contextual information ensures that the outputs are both relevant and meaningful.

## 3. PDFPlumber and BeautifulSoup: For extracting text from PDFs and online documentation

- **PDFPlumber**: This library specializes in extracting structured data from PDFs, including text, tables, and metadata. It ensures robust handling of PDFs with complex layouts or formats.
- **BeautifulSoup**: A Python library for web scraping, it allows for the efficient extraction of data from online documentation or HTML content. It helps parse and navigate HTML or XML documents to extract useful information for downstream processing.

Together, these tools enable the system to gather raw textual data from diverse sources, making it available for analysis and solution generation.

## 4. Vector Database (Milvus): For Storing and Retrieving Vector Embeddings

- **Milvus** is a high-performance vector database optimized for managing and querying vector embeddings. In this system:
- Textual data is transformed into numerical vector representations using embedding models.
- These vectors are stored in **Milvus**, enabling efficient similarity searches.
- When users input a query, **Milvus** retrieves the most relevant data by comparing embeddings, ensuring fast and accurate results.
-

### 5. Text Embedding Models: For creating vector representations of textual data

Text embedding models convert textual data into high-dimensional vector representations, capturing semantic meaning and relationships. These embeddings are crucial for:

- Storing data in a way that allows similarity comparisons.
- Enabling retrieval of contextually similar content from the vector database. Popular models like OpenAI's embedding models or other transformer-based architectures might be used here.

## Problem Statement

**Objective**: To develop an intelligent and automated system for diagnosing, exploring, and resolving common errors faced by Windows 11 users.

**Current Challenge**: Identifying and resolving errors such as blue screens, update failures, and performance issues often requires users to search through extensive forums, documentation, and blogs. This process is not only time-consuming but also challenging for non-technical users.

**Data Complexity**: The diverse nature of error-related data, including logs, troubleshooting guides, and community FAQs in multiple formats (PDFs, web pages, and databases), complicates efficient retrieval and resolution.

**Goal**: Create a centralized platform that integrates:

- **Data Storage**: Efficiently process, store, and manage structured and unstructured troubleshooting data.
- **Error Resolution Workflow**: Leverage RAG-based systems to index, retrieve, and generate contextualized solutions to user queries.
- **User Experience**: Provide an intuitive interface that simplifies querying, error exploration, and the resolution process.

## Goals

An intelligent error resolution platform that revolutionizes how users interact with Windows 11 troubleshooting and diagnostics:

1. **Automated Data Pipeline**:
   a. Automated ingestion and processing of error logs, troubleshooting guides, and FAQs from official Microsoft documentation and community forums.
   b. Systematic storage of logs, solutions, and metadata in a robust cloud infrastructure.
   c. Efficient data synchronization and update mechanisms to ensure the system stays current.
2. **Intelligent Error Processing**:
   a. Advanced text extraction and processing to handle diverse data formats such as PDFs, web pages, and structured logs.
   b. Vector embeddings to enable precise semantic search capabilities.
   c. Multi-modal Retrieval-Augmented Generation (RAG) system for contextual understanding and error resolution.
3. **Interactive Troubleshooting Interface**:
   a. User-friendly platform with intuitive query inputs for natural language or error code-based searches.
   b. Real-time generation of step-by-step troubleshooting instructions and contextual solutions.
   c. Error history tracking and insights to assist users in resolving recurring issues.
4. **Knowledge Base Development**:
   a. Incremental building of an error resolution knowledge base with validated solutions linked to their source documentation.
   b. Cross-error pattern analysis to generate insights and improve recommendations.
   c. Real-time updates and integration with official Microsoft resources to ensure the accuracy and relevance of solutions.

## Proof of Concept

1. **Data Ingestion Pipeline**:
   a. Successfully implemented an automated data ingestion pipeline for collecting error-related data, including:

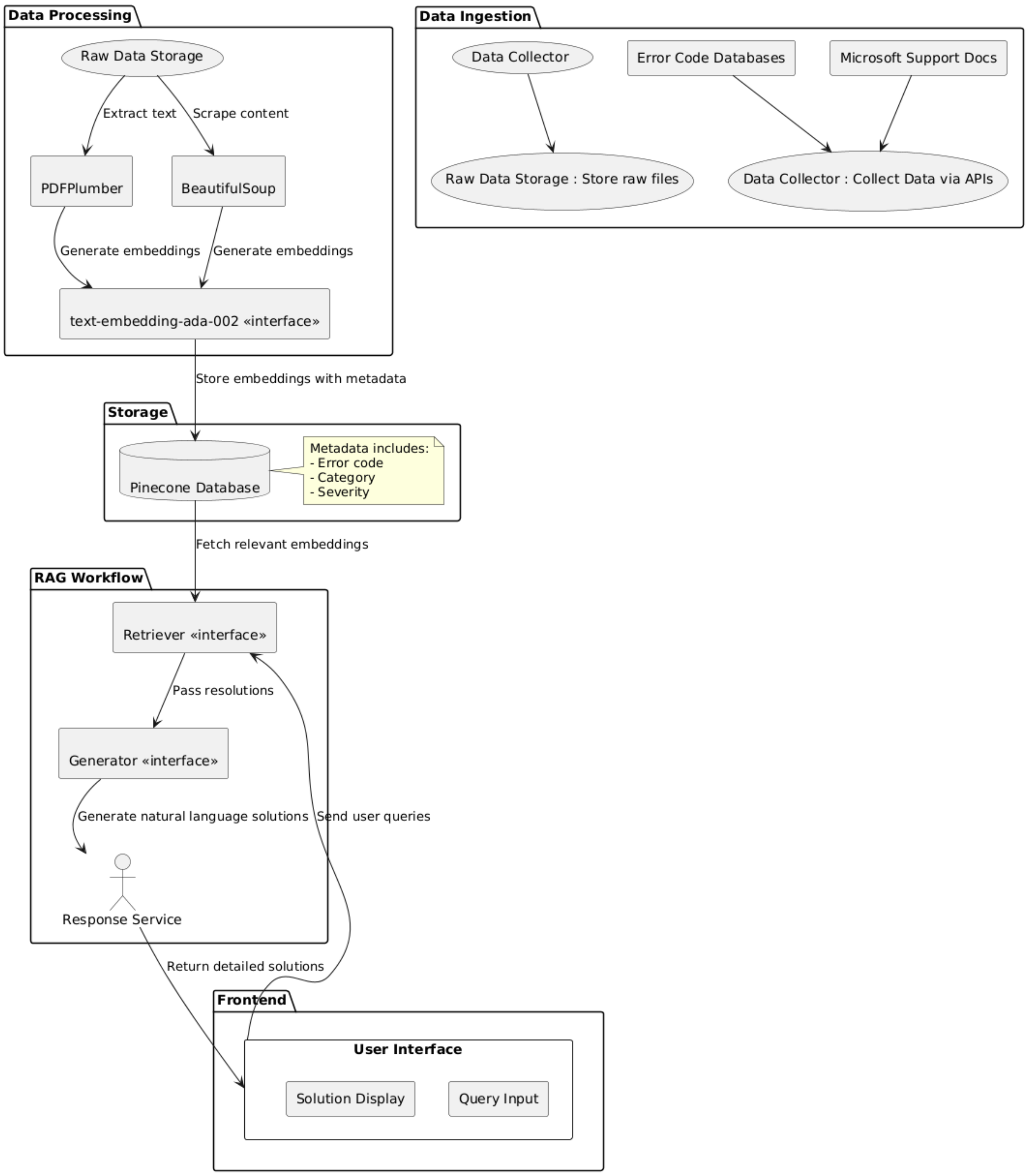
        i. Official Microsoft support documentation.
        ii. Error code databases and troubleshooting guides.
        iii. Community forum discussions.
    b. Utilized tools like Airflow to orchestrate workflows, ensuring seamless extraction, transformation, and loading of data into the system.
    c. Raw data (PDFs, HTML pages, and logs) is systematically stored in a cloud-based infrastructure for further processing.

2. **Error Processing Pipeline**:
    a. Developed a robust pipeline to process error-related documents through multiple stages:
        i. Extracted structured and unstructured data using PDFPlumber and BeautifulSoup.
        ii. Generated vector embeddings of processed content using models like `text-embedding-ada-002`.
        iii. Stored embeddings in a Pinecone vector database with metadata for efficient retrieval and contextual analysis.
    b. Ensured end-to-end integration between data processing components to enable real-time querying and troubleshooting.

3. **Interactive Troubleshooting System**:
    a. Implemented an interactive user interface with the following capabilities:
        i. Real-time natural language query processing to address user-specific issues.
        ii. Integration of a RAG workflow to fetch and generate accurate solutions based on retrieved embeddings and contextual data.
        iii. Step-by-step troubleshooting responses generated dynamically using GPT-based LLMs.
    b. Incorporated an error history tracking feature for users to revisit previous solutions and analyze recurring issues.

4. **Knowledge Base System**:
    a. Successfully integrated a knowledge base with Pinecone to store validated solutions linked to their respective source documents.
    b. Incrementally updated the knowledge base with new insights and solutions, ensuring the system evolves continuously.
    c. Developed mechanisms for cross-error pattern analysis to improve the quality of recommendations and enhance user satisfaction.

## Systems Architecture Diagram



1. **Data Ingestion**:
   a. Collect error logs, FAQs, and troubleshooting guides from:
      i. Microsoft Support documentation.
      ii. Error code databases.
2. **Data Processing**:
   a. Extract text using PDFPlumber and BeautifulSoup.

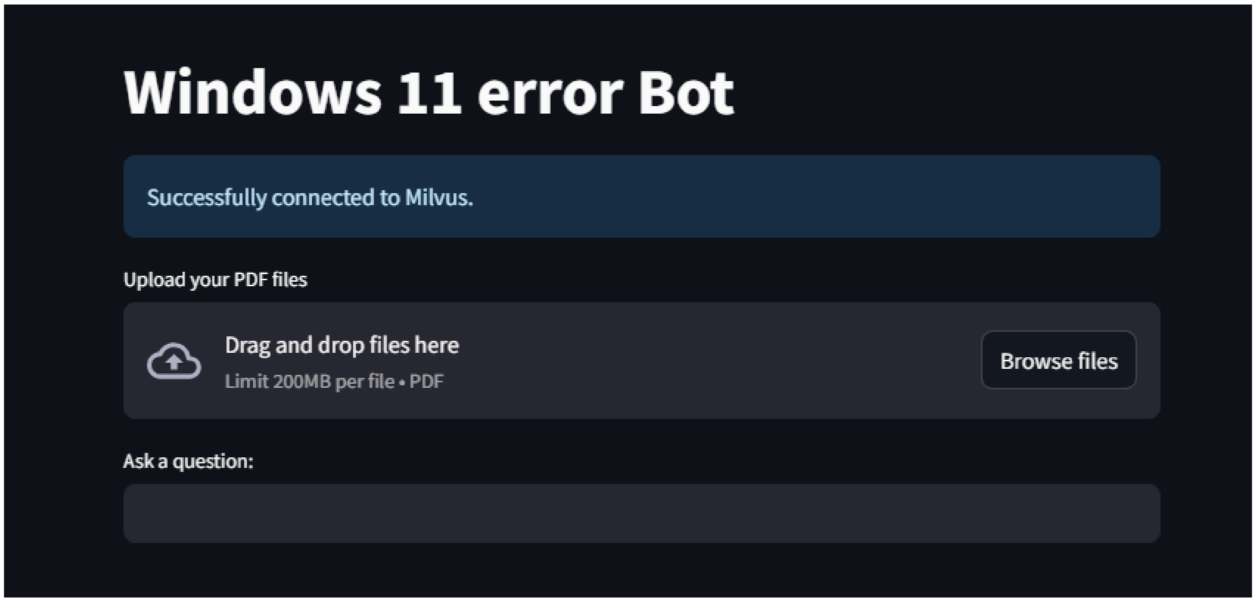
    b. Generate vector embeddings using models like text-embedding-ada-002.
3. **Storage**:
    a. Store vector embeddings in Pinecone with metadata tags (error code, category, severity).
4. **RAG Workflow**:
    a. **Retrieve**: Fetch relevant error resolutions from the vector database.
    b. **Generate**: Use an LLM to generate natural language responses.
5. **Frontend**:
    a. Desktop or web-based UI with query input and detailed solution display.

## Walkthrough of the Application

This app is designed to extract and analyze error details from PDF files related to Windows 11 issues. It connects seamlessly to the **Milvus** vector database for storing and querying error reports for future reference. Below is a walkthrough of the app's key functionality:

**Milvus Connection:**

- Upon starting, the app establishes a connection with the **Milvus** vector database. A success message confirms that the connection is active and ready for operations.



Windows 11 error Bot

Successfully connected to Milvus.

Upload your PDF files

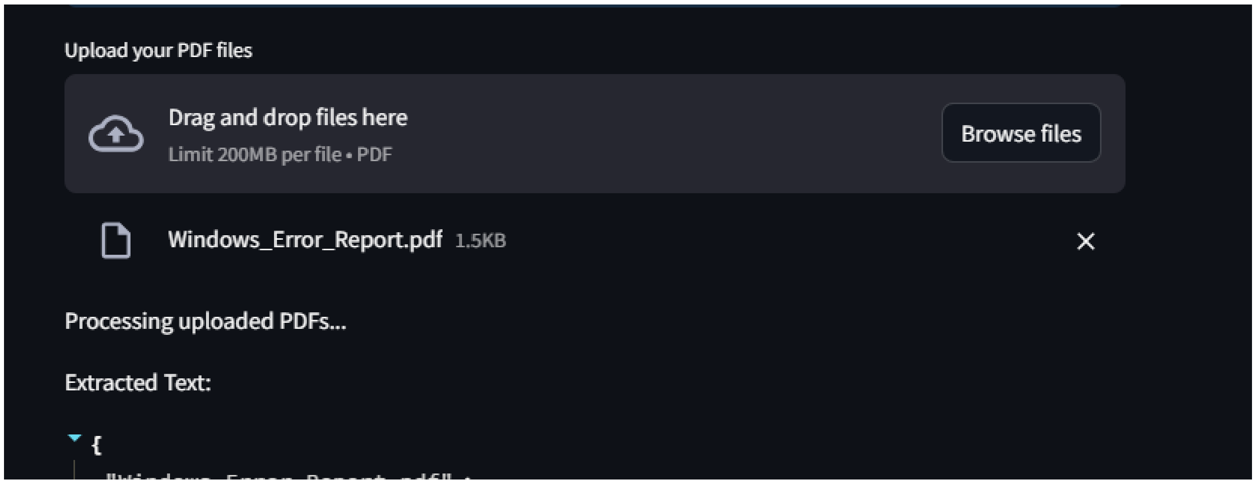
Drag and drop files here

Limit 200MB per file • PDF

Ask a question:

**PDF Upload and Text Extraction:**

- Users can upload PDF files by dragging and dropping them into the interface or using the file browser.
- Once uploaded, the app processes the PDFs and extracts the embedded text, displaying it in a structured format for the user to verify.


Upload your PDF files

Drag and drop files here
Limit 200MB per file • PDF

Windows_Error_Report.pdf 1.5KB

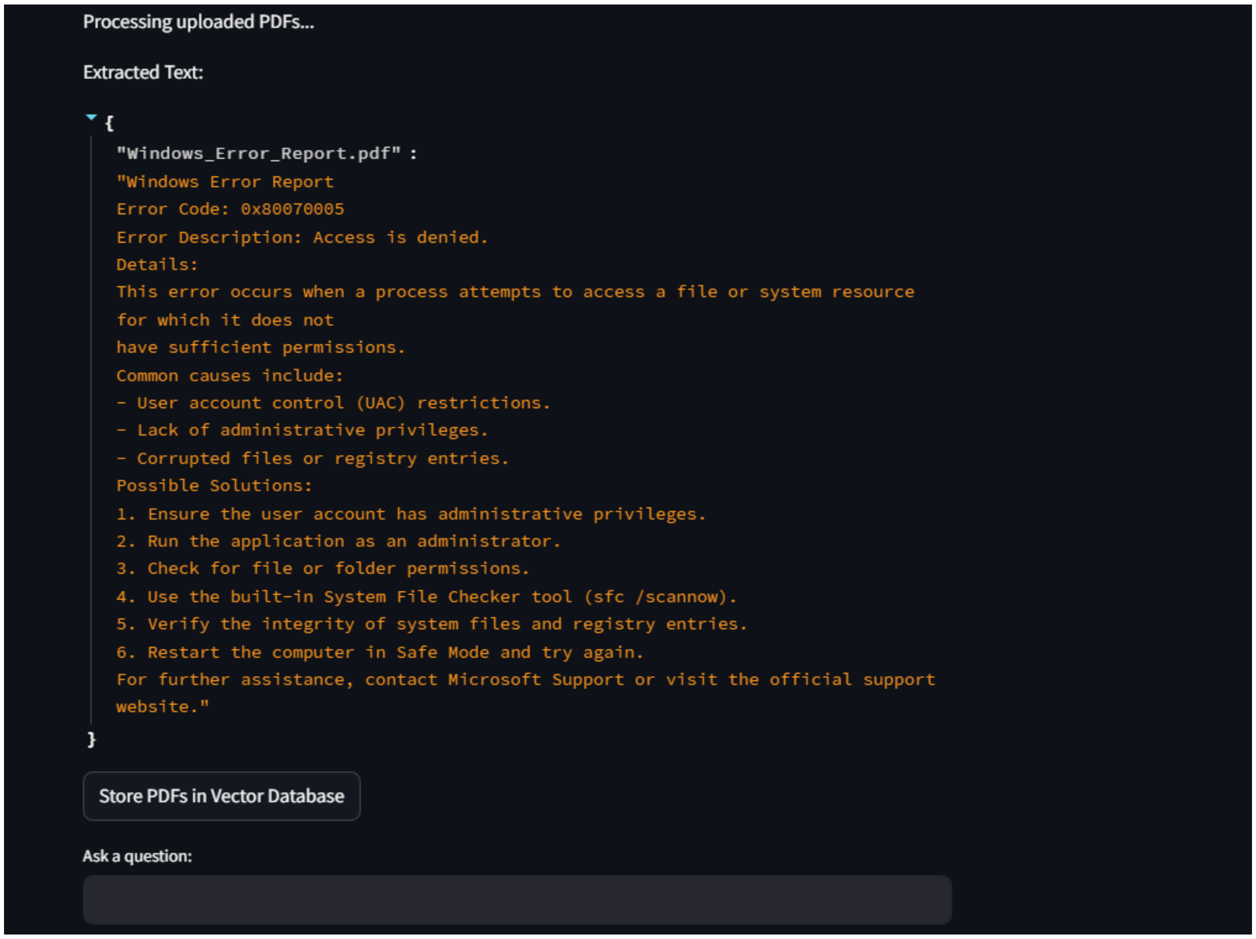
Processing uploaded PDFs...

Extracted Text:

```
{
```

**Text Extraction Example:**

- For a sample file `Windows_Error_Report.pdf`, the app extracted key details such as error codes, descriptions, potential causes, and recommended solutions, presenting them clearly in the extracted text section


Processing uploaded PDFs...

Extracted Text:

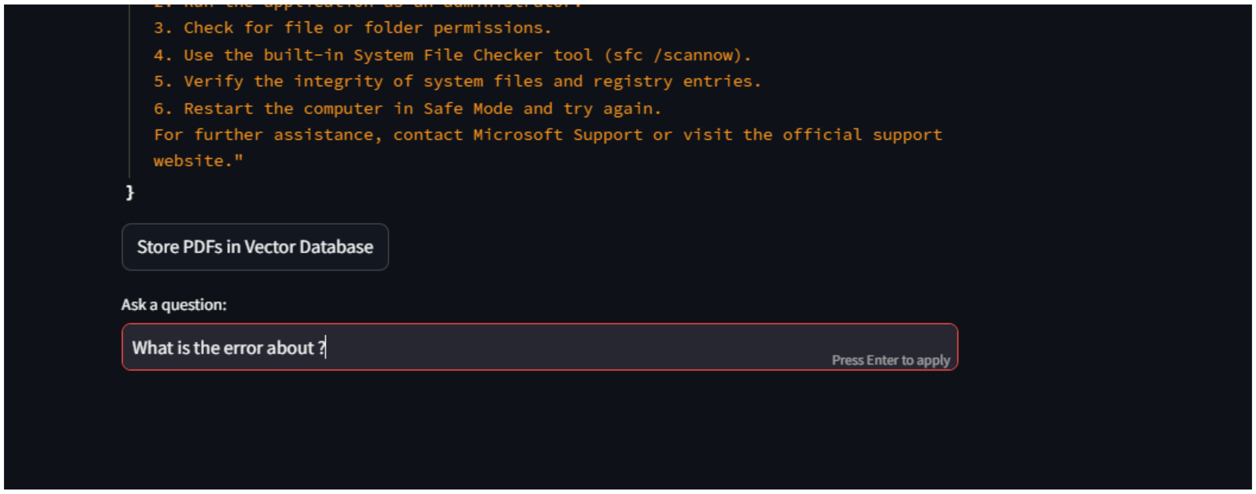
```
{
  "Windows_Error_Report.pdf" :
  "Windows Error Report
  Error Code: 0x80070005
  Error Description: Access is denied.
  Details:
  This error occurs when a process attempts to access a file or system resource
  for which it does not
  have sufficient permissions.
  Common causes include:
  - User account control (UAC) restrictions.
  - Lack of administrative privileges.
  - Corrupted files or registry entries.
  Possible Solutions:
  1. Ensure the user account has administrative privileges.
  2. Run the application as an administrator.
  3. Check for file or folder permissions.
  4. Use the built-in System File Checker tool (sfc /scannow).
  5. Verify the integrity of system files and registry entries.
  6. Restart the computer in Safe Mode and try again.
  For further assistance, contact Microsoft Support or visit the official support
  website."
}
```

**Storing Data in Vector Database:**

- After text extraction, users can store the data in the **Milvus** vector database by clicking the "Store PDFs in Vector Database" button. This enables efficient storage and future retrieval based on similarity queries.

**Query Feature:**

- The app includes a query input field where users can ask questions about the stored error reports. For instance, typing "What is the error about?" will search for the most relevant information in the database and provide insights.



```
    3. Check for file or folder permissions.
    4. Use the built-in System File Checker tool (sfc /scannow).
    5. Verify the integrity of system files and registry entries.
    6. Restart the computer in Safe Mode and try again.
    For further assistance, contact Microsoft Support or visit the official support
    website."
}
```

Ask a question:

What is the error about ?

## FastAPI for Backend Services

FastAPI serves as the backbone of the application, delivering high-performance APIs for error diagnostics, troubleshooting, and user interactions:

1. **Error Exploration API**:
    a. Enables users to browse metadata associated with error codes, logs, and solutions through a user-friendly query system.
    b. Allows selection of error details via dropdown menus or by entering error codes and descriptions.
2. **Dynamic Error Resolutions**:
    a. Provides on-demand step-by-step solutions to errors using a combination of retrieval (Pinecone) and generation (GPT-4).
    b. Integrates context-aware troubleshooting steps from official documentation and validated sources.
3. **Multi-Modal Retrieval (RAG)**:
    a. Uses advanced embedding models (e.g., `text-embedding-ada-002`) for semantic search and Pinecone for efficient vector retrieval.

    b. Employs an LLM-based pipeline to combine retrieved data and generate concise, relevant solutions.

4. **Q&A Interface**:
    a. Supports natural language questions, allowing users to ask queries such as:
        i. "How do I fix Windows Update error 0x80070005?"
        ii. "What causes the blue screen error with code 0x00000116?"
    b. Retrieves contextual answers from embeddings in Pinecone, delivering solutions in an intuitive, conversational format.
5. **Error Report and Insights Generation**:
    a. Generates detailed, link-embedded error resolution reports for end-users or IT professionals.
    b. Stores verified solutions and troubleshooting steps as searchable insights for future reference.

## Deployment of the Workflow

The application is fully containerized and deployed on a cloud platform using Docker, ensuring high scalability, accessibility, and reliability:

1. **Containerization**:
    a. Both the FastAPI backend and the Streamlit-based frontend are containerized using Docker.
    b. Docker Compose orchestrates their interactions for seamless functionality.
2. **Public Accessibility**:
    a. Deployed on a cloud platform (e.g., AWS or GCP), enabling secure and remote access to both the API and frontend.
    b. Supports encrypted connections and secure APIs for sensitive data handling.
3. **Scalability**:
    a. The cloud platform dynamically scales the application to handle increasing user loads and high query volumes.
    b. Ensures consistent performance and availability during peak usage.

## Summary of the Application Workflow

1. **Data Ingestion**:

   a. The pipeline collects error logs, troubleshooting guides, and FAQs from Microsoft documentation and community forums.
   b. Raw data is stored in cloud infrastructure for processing and indexing.
2. **Backend Services**:
   a. FastAPI processes error exploration queries, provides on-demand troubleshooting instructions, powers Q&A interactions, and generates error resolution reports.
3. **User Interaction with Streamlit**:
   a. The frontend enables users to:
      i. Enter error codes or questions.
      ii. Receive step-by-step solutions.
      iii. View previously saved solutions and interact with the Q&A bot for contextual insights.
4. **Error Knowledge Base**:
   a. Pinecone indexes error logs and validated solutions, enabling semantic search capabilities.
   b. Users can search and explore previously indexed error resolutions differentiated by source and relevance.

## Contributors

- Nagapriyaham Pindi (50%)
- Vishodhan Krishnan (50%)

## References

- FastAPI Documentation
- Streamlit Documentation
- Multimodal RAG Slide Deck Example
- GPT-4 API Documentation

- Airflow Documentation