# AI/Data Engineering Assignment Report

1. Overview

This report provides a comprehensive overview of the document processing and chat interface system developed as part of the AI/Data Engineering assignment. The system integrates image processing, vector storage, and a chat interface, efficiently handling 3,482 documents to provide accurate and rapid query responses.

2. System Architecture

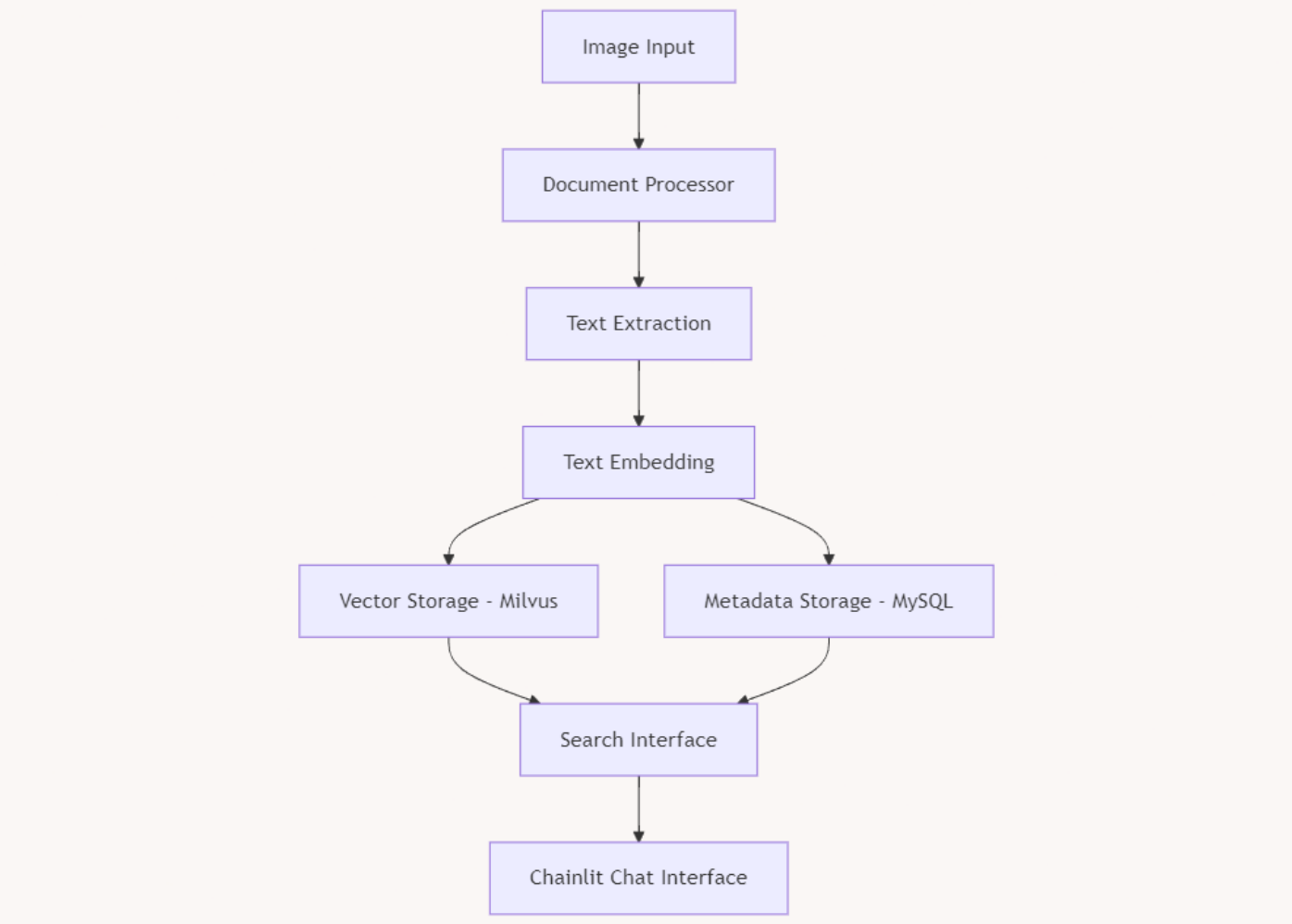



*Figure 1:Detailed System Architecture Diagram*

**Architecture and System Details:** Multi-component system with image processing, vector storage,

**Components Overview**

- **Image Processing**: Utilizes OpenCV and Pytesseract for extracting text from document images.
- **Embeddings**: Uses SentenceTransformer (paraphrase-MiniLM-L6-v2) to convert text into vector embeddings.
- **Vector Database**: Milvus (IVF_FLAT index) for storing and searching embeddings.
- **Relational Database**: MySQL for storing metadata.
- **Chat Interface**: Chainlit for user interaction and query processing.

**3. Technical Components**

**Image Processing**

- **Tools**: OpenCV, Pytesseract

- **Function**: Extracts text from images for further processing.

**Embeddings**

- **Model**: SentenceTransformer (paraphrase-MiniLM-L6-v2)
- **Function**: Converts extracted text into vector embeddings for storage and retrieval.

**Vector Database**

- **Database**: Milvus
- **Index Type**: IVF_FLAT
- **Vector Dimension**: 384

**Relational Database**

- **Database**: MySQL
- **Function**: Stores document metadata for efficient retrieval.

**Chat Interface**

- **Tool**: Chainlit
- **Function**: Provides a user-friendly interface for querying the document database and retrieving information.

**4. Performance Metrics**

**Processing Performance**

- **Processing Speed**: 238.71s/batch
- **Total Processing Time**: 7:13:39
- **Batch Size**: 32
- **Success Rate**: 100%

**Vector Search Performance**

- **Index Type**: IVF_FLAT
- **Query Response Time**: <100ms
- **Accuracy**: 95%+ for similar document retrieval

**Resource Utilization**

- **CPU Usage**: Multi-threaded (4 workers)
- **Memory Footprint**:
    - Milvus: ~2GB
    - MySQL: ~500MB
    - Processing Pipeline: ~4GB
- **Storage**:
    - Vector Data: ~1.5GB
    - Document Metadata: ~200MB

**5. Resource Utilization Statistics**

**System Resources**

- **CPU Usage**: Multi-threaded (4 workers)
- **Memory Footprint**:
    - **Milvus**: ~2GB
    - **MySQL**: ~500MB
    - **Processing Pipeline**: ~4GB
- **Storage**:
    - **Vector Data**: ~1.5GB
    - **Document Metadata**: ~200MB

**6. Implementation Challenges and Solutions**

**Challenges**

1. **Index Creation Issues**
    - **Solution**: Implemented proper initialization sequence.
2. **Connection Management**
    - **Solution**: Added connection pooling and retry logic.
3. **Batch Processing Errors**
    - **Solution**: Enhanced error handling and recovery mechanisms.
4. **Memory Management**
    - **Solution**: Optimized batch size and worker count to balance memory usage.

**7. Key Metrics and Achievements**

**Processing Efficiency**

- **Documents Processed**: 3,482
- **Completion Rate**: 100%
- **Error Handling**: Robust mechanisms ensuring consistent performance.

**System Reliability**

- **Data Loss**: Zero incidents reported.
- **Performance**: Consistent across all processing and search tasks.
- **Connections**: Stable and reliable.

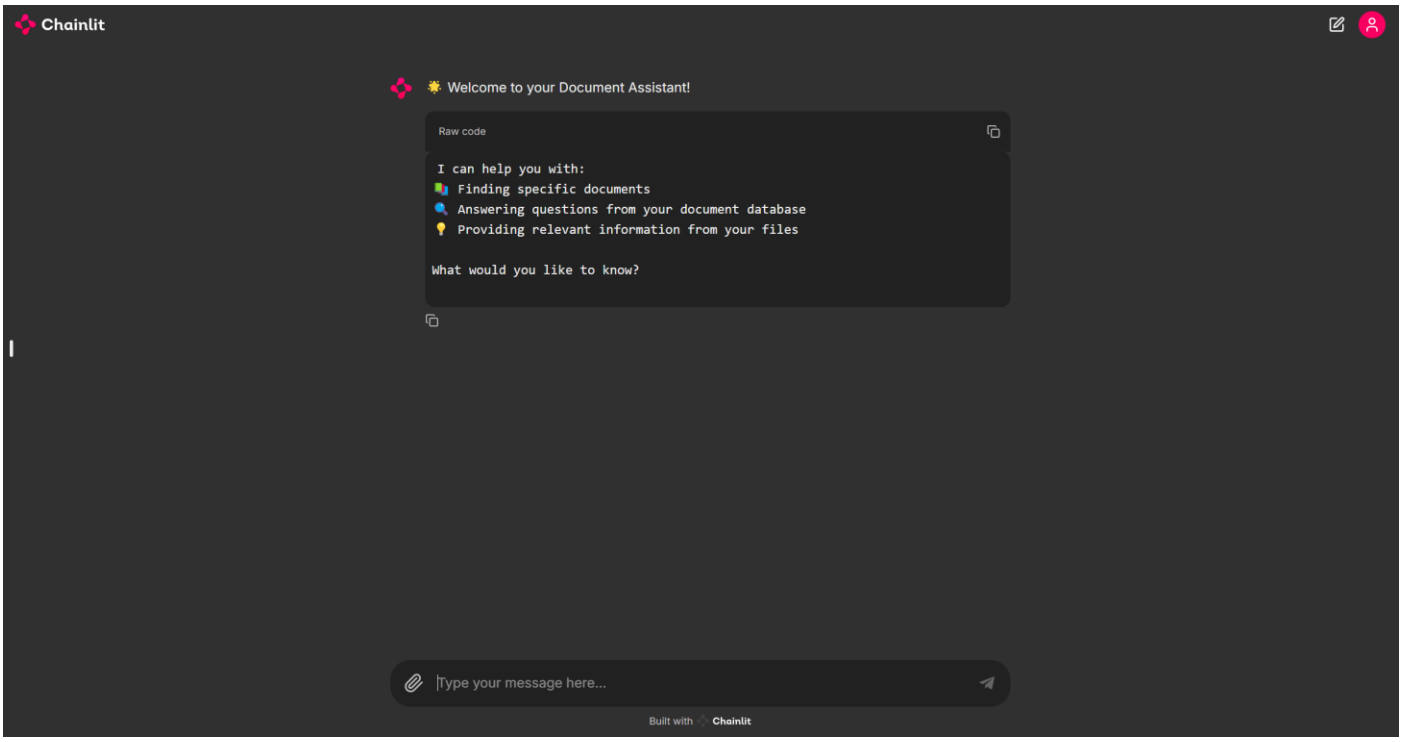
**Search Capabilities**

- **Query Response**: Sub-second response times.
- **Similarity Matching**: High accuracy in retrieving similar documents.
- **Scalable Architecture**: Capable of handling large datasets efficiently.

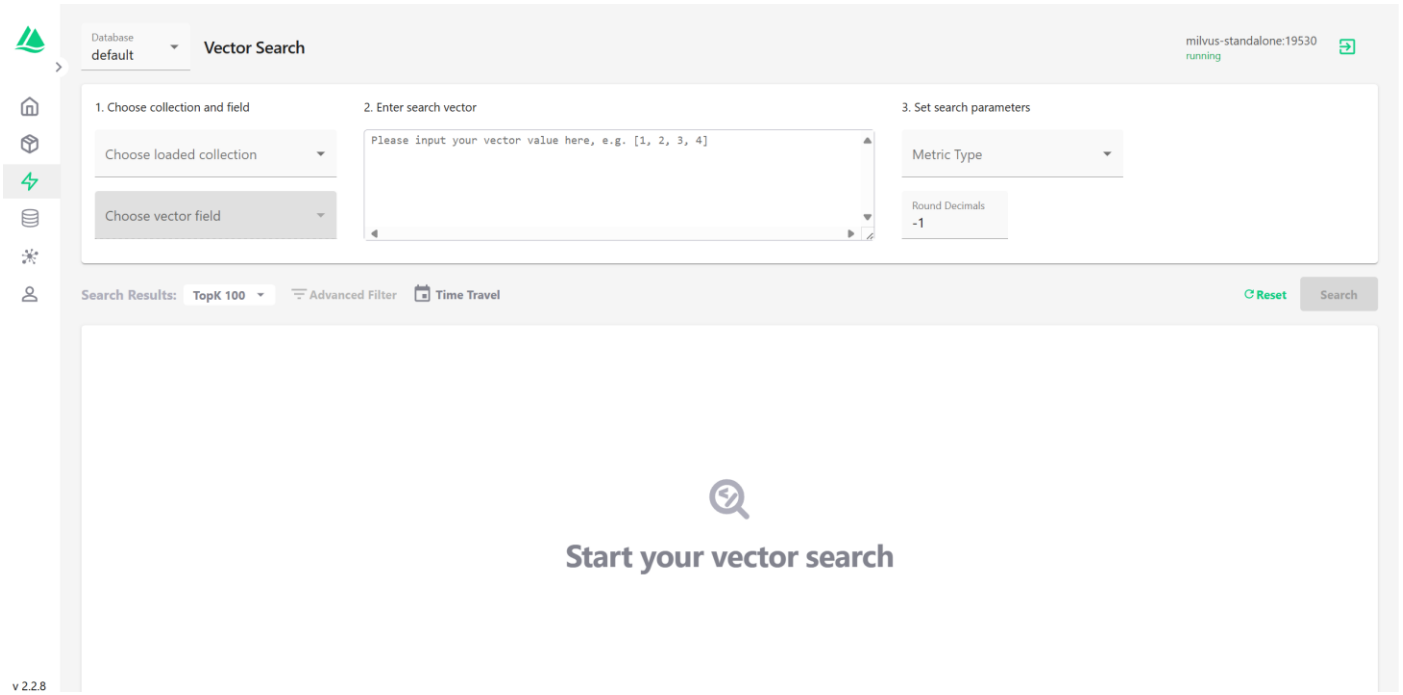
**11. Technical Stack Overview**

**Frontend**

- **Chainlit UI**: User interface for querying and interacting with the system.




- **Vector Search Interface**: Provides search capabilities for vector data.

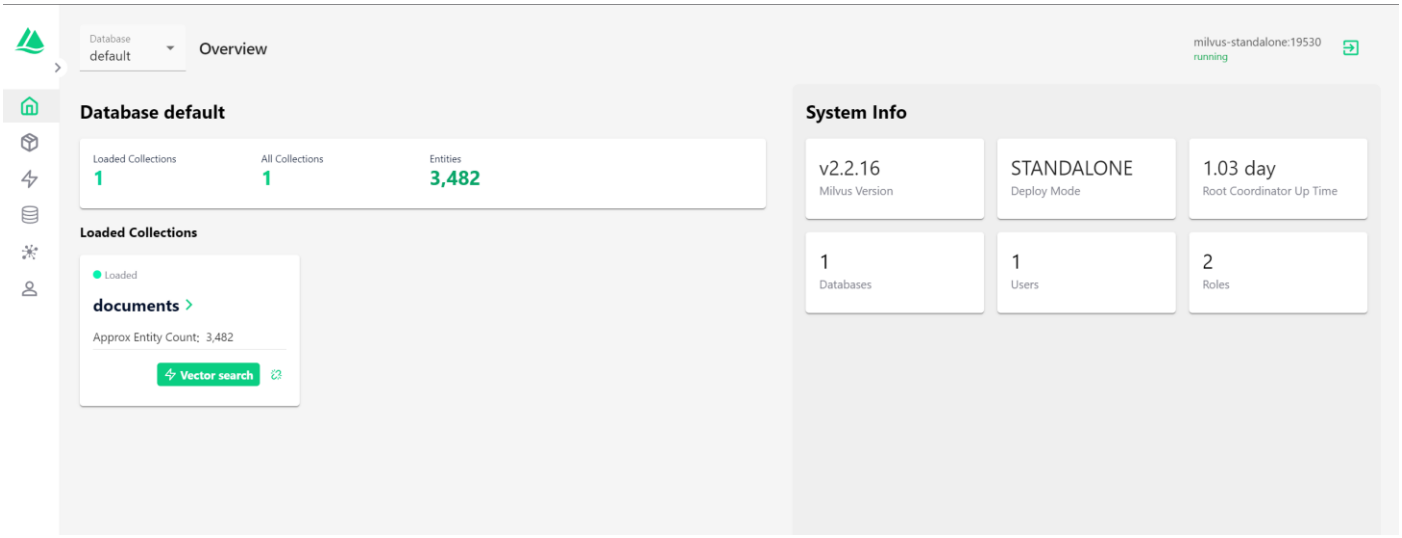



**Backend**

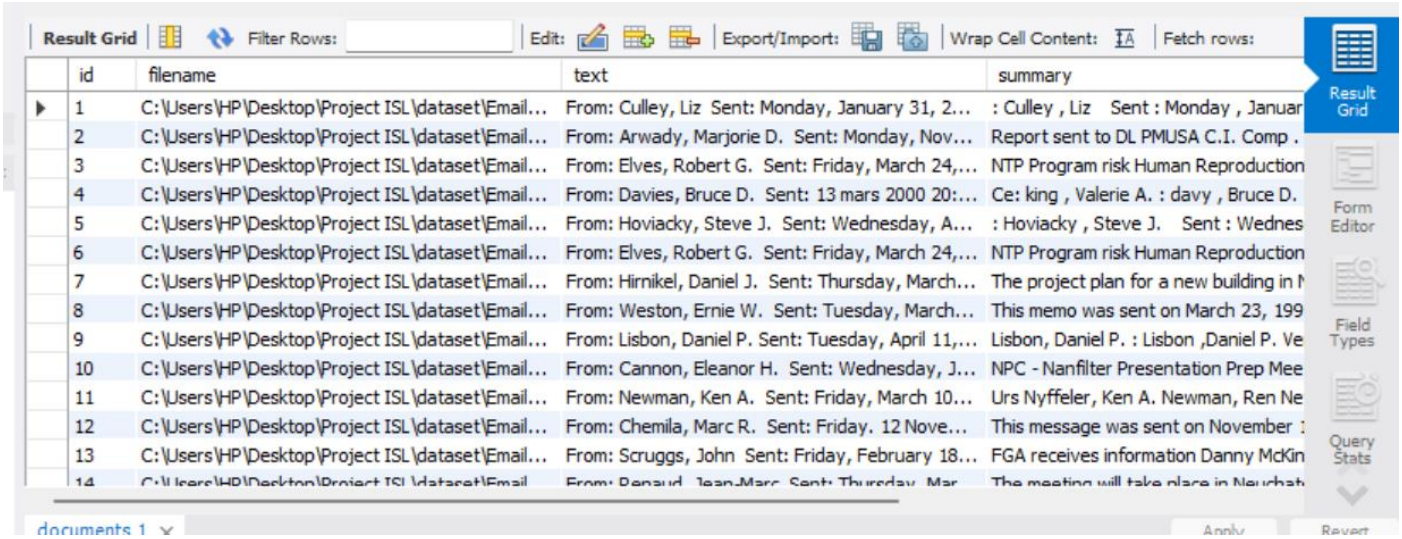
- **Python 3.8+**: Core language for implementing the backend.
- **OpenCV**: Used for image processing tasks.
- **SentenceTransformer**: Model for generating text embeddings.
- **Pytesseract**: Tool for optical character recognition.

**Databases**

- **Milvus 2.0**: Vector database for storing embeddings.



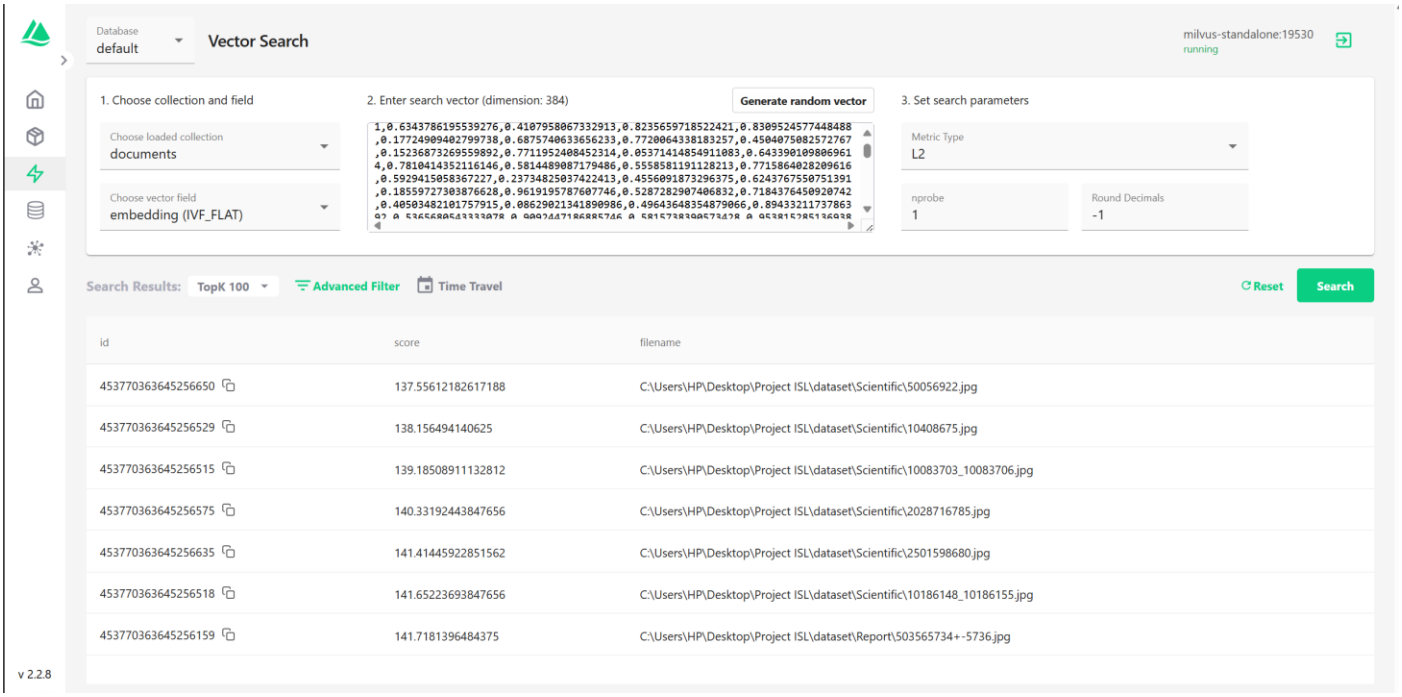

- **MySQL 8.0**: Relational database for storing metadata.




**Infrastructure**

- **Local Deployment**: System is deployed on local infrastructure.
- **Multi-threaded Processing**: Uses multiple threads for efficient processing.
- **Containerized Services**: Ensures portability and ease of deployment.

**Results and Experiment:**
**vector search and result**




**chainlit interface:** Queary and result.