Compliance Validation System Documentation

Overview

The Compliance Validation System is designed to streamline the process of defining, managing, and validating compliance pointers. The system integrates MongoDB for data storage, FAISS for semantic embedding and vector search, and OpenAI GPT for compliance analysis. Users can define pointers, upload supporting documents, and validate compliance status using machine learning techniques.

System Components

1. Root Directory Files

1.1 compliance_operations.py

This file manages compliance result operations, including adding, retrieving, updating, and deleting compliance results in the database.

1.2 db_connection.py

Handles MongoDB connection using pymongo. It caches the database connection and securely accesses MongoDB credentials from Streamlit secrets.

1.3 document_operations.py

Provides CRUD operations for documents, including linking documents to specific compliance pointers and handling bulk deletions.

1.4 Landing_Page.py

The starting point for the application, allowing users to navigate to different sections using a sidebar or proceed with compliance setup.

1.5 pointer_operations.py

Manages CRUD operations for compliance pointers, which include information like objectives, compliance requirements, and supporting document points.

2. FAISS Embedding Generation

generate_embeddings.py

This script processes documents (e.g., PDFs) into semantic embeddings using LangChain, OpenAI embeddings, and FAISS. It:

- Splits document text into meaningful chunks.
- Converts the chunks into embeddings.
- Stores these embeddings in a FAISS index file for later retrieval during compliance checks.

---

3. Streamlit Pages

3.1 1_Year_Selection.py

Allows users to select a compliance year and proceed to define pointers.

3.2 2_Pointer_Definition.py

Enables users to:

- Define or edit pointers.
- Add objectives, compliance requirements, and supporting document points.
- Upload supporting documents.

3.3 3_Compliance_Analysis.py

Executes the compliance validation process by:

- Extracting text from uploaded documents using OpenAI Vision.
- Matching extracted content against compliance requirements using a language-specific FAISS index.
- Generating a compliance status (e.g., Fully Compliant, Partially Compliant, Not Compliant) using OpenAI GPT.

3.4 4_View_Pointers.py

Lists saved pointers, their compliance statuses, uploaded documents, and options to edit, delete, or analyze compliance.

How to Set Up MongoDB

Step 1: Download and Install MongoDB

1. Visit the MongoDB Community Edition Download Page.
2. Select your operating system (Windows, macOS, or Linux).
3. Download the **MongoDB Installer** for your OS.
4. Run the installer and follow the prompts to install MongoDB.

Step 2: Install MongoDB Compass

1. During installation, ensure the checkbox to install MongoDB Compass is selected.
2. MongoDB Compass is a GUI tool for managing your database. If not selected during installation, you can download it separately from MongoDB Compass Download Page.

Step 3: Run MongoDB

1. Start MongoDB server by executing the mongod command in your terminal (if installed locally).
2. Open MongoDB Compass.

Step 4: Create a Database and Collections in MongoDB Compass

1. Launch MongoDB Compass and connect to your MongoDB instance (e.g., mongodb://localhost:27017).
2. On the left sidebar, click on **"Create Database"**.
3. Name the database: **compliance_db**.
4. Create three collections within the database:
    - **pointers**
    - **documents**
    - **compliance_results**

Streamlit Application Setup

Step 1: Install Dependencies

1. Clone the repository containing the application code.
2. Create a Python virtual environment and activate it:

```
python -m venv venv # On Windows,

source venv/bin/activate  # On MAC,
```

3. Install dependencies:

```
pip install -r requirements.txt
```

Key Functionalities

Pointer Definition

1. Define pointers with objectives, compliance requirements, and supporting document points.
2. Upload supporting documents (PDFs, images, etc.).

Compliance Analysis

1. Validate uploaded documents against compliance requirements.
2. Use FAISS and GPT to evaluate compliance status.

Pointer Management

1. View saved pointers.
2. Edit, delete, or reanalyze pointers.

Database Schema

1. pointers

Stores compliance pointers with details like objectives, requirements, and document metadata.

```
{

  "_id": "ObjectId",

  "name": "string",

  "objective": "string",

  "compliance_requirements": "string",

  "supporting_document_points": "string",
  "language": "string",
  "year": "integer",
  "compliance_status": "string"
}
```

2. documents

Stores supporting documents linked to pointers.

```
{
  "_id": "ObjectId",
  "pointer_id": "ObjectId",
  "document_name": "string",
  "document_data": "binary",
  "upload_date": "datetime"
}
```

3. compliance_results

Stores compliance validation results.

```
{
  "_id": "ObjectId",
  "pointer_id": "ObjectId",
  "compliance_status": "string",
  "details": "string",
  "checked_date": "datetime"
}
```

Best Practices

1. **Secure API Keys:** Keep sensitive keys like OPENAI_API_KEY in a .env file.
2. **Regular Backups:** Back up the MongoDB database to prevent data loss.
3. **Test Vector Stores:** Validate the FAISS indexes regularly to ensure semantic embeddings work as expected.