User Guide for Operating and Maintaining the RAG-PDF Summarizer and Q/A Model

---

1. Operating the System

Step 1: Starting the Application

Ensure all dependencies are installed. Run the following commands in your terminal:
bash
Copy code

```
pip install streamlit PyPDF2 langchain-community faiss-cpu google-generativeai
python-dotenv
```

1.
    - Use `faiss-gpu` if your system supports CUDA.

Start the application:
bash
Copy code

```
streamlit run app.py
```

2.

Step 2: Uploading PDF Files

1. Navigate to the **Sidebar Menu** in the Streamlit app.
2. Click on **Upload PDF Files**.
3. Drag and drop multiple PDFs or click to upload files from your system.
4. Click **Submit & Process** to start processing.

Step 3: Asking Questions

1. In the **Main Interface**, type your question in the input box.
2. The system retrieves relevant information from the processed PDFs and provides an answer.
3. Repeat this step to ask additional questions.

---

2. Maintaining the System

Regular Maintenance Tasks

1. **Dependency Updates**:

Periodically update libraries to ensure compatibility:
bash
Copy code
```
pip install --upgrade streamlit PyPDF2 langchain-community faiss-cpu google-generativeai
```

    ○

2. **Clearing Temporary Files**:

Remove unnecessary files in the temporary directory (`/tmp`) or any local vector stores (`vectorstore` directory) to save space:
bash
Copy code
```
rm -rf /tmp/*
rm -rf vectorstore/*
```

    ○

3. **Backup Vector Stores**:
    ○ Backup the `vectorstore` directory regularly to avoid re-processing PDFs.
4. **Environment Variable Management**:

Store API keys securely in a `.env` file. Ensure this file is not shared publicly:
text
Copy code
```
GOOGLE_API_KEY=<your_google_api_key>
```

    ○

Performance Optimization

- If processing is slow, optimize by:

Adjusting the chunk size in the `get_text_chunks()` function:
python
Copy code
```
text_splitter = RecursiveCharacterTextSplitter(chunk_size=5000, chunk_overlap=500)
```

1.
2. Upgrading to a GPU-enabled system for faster FAISS operations.

---

3. Troubleshooting Common Errors

Error 1: Missing Dependencies

- *Symptom*: "Module not found" error for `langchain_community.vectorstores` or `faiss`.

Fix: Install missing dependencies using:
bash
Copy code
```
pip install langchain-community faiss-cpu
```

-

Error 2: FAISS Runtime Error

- *Symptom*: `RuntimeError` related to file access for `index.faiss`.

- *Fix*: Ensure the `vectorstore` directory exists and contains the `index.faiss` file.

Error 3: Unsafe Deserialization Warning

- *Symptom*: `ValueError` about dangerous deserialization.
- *Fix*: Use `allow_dangerous_deserialization=True` only for trusted files. Avoid using external unverified data.

Error 4: Google API Key Issues

- *Symptom*: Errors when generating embeddings.
- *Fix*: Verify your API key in the `.env` file and ensure it is active in your Google Cloud account.

Error 5: Missing Vector Store

- *Symptom*: Warning about missing `index.faiss`.
- *Fix*: Reprocess PDFs to recreate the vector store.

---

4. Advanced Maintenance

Switching Between Local and Cloud Storage

- To use cloud storage for vector files (e.g., AWS S3):

Upload processed files to the cloud:
bash
Copy code

```
aws s3 cp vectorstore/ s3://your-bucket-name/vectorstore/ --recursive
```

1.
2. Modify the `load_vectorstore` function to fetch files from the cloud.

Improving Security

- Avoid storing sensitive data in code. Use environment variables or secret managers (e.g., AWS Secrets Manager, HashiCorp Vault).

---

5. Future Improvements

Suggestions for Better Usability

1. **Interactive Logs**:
    - Add a log viewer in the Streamlit app to track processing steps.
2. **Error Reporting**:
    - Automate error reporting with detailed logs saved in a separate directory.
3. **Cloud Integration**:
    - Allow direct PDF uploads from cloud storage (e.g., Google Drive or AWS S3).