Refund Policy Q&A Application Documentation

1. Overview

The **Refund Policy Q&A** application is a Streamlit-based interactive tool that allows users to upload a PDF document containing their company refund policies and then ask questions about its content. The system uses Natural Language Processing (NLP) techniques leveraging LangChain, HuggingFace embeddings, and a transformer-based language model (Google Flan-T5) to provide relevant answers extracted from the uploaded document.

---

2. Features

- Upload company policy PDF files
- Extract text from PDF pages automatically
- Split text into chunks for better semantic search
- Create or load a cached FAISS vector store to speed up embedding search
- Use sentence-transformer embeddings for document indexing
- Query the document with natural language questions
- Get relevant answers generated by a language model

---

3. Installation & Setup

Requirements

- Python 3.7+
- Streamlit
- PyPDF2
- LangChain
- FAISS
- HuggingFace transformers and sentence-transformers
- Additional dependencies (listed in requirements.txt)

Install dependencies

```bash
pip install streamlit PyPDF2 langchain faiss-cpu transformers sentence-transformers
```

Running the app

```bash
streamlit run app.py
```

Open your browser and navigate to the URL shown (usually http://localhost:8501).

---

4. Code Explanation

Key components:

- **PDF Upload & Extraction**

  Uses st.file_uploader to accept PDF files, and PyPDF2 to extract text from each page.

- **Text Splitting**

  Uses LangChain's CharacterTextSplitter to break large text into manageable chunks (1000 chars with 200 overlap).

- **Embeddings**

  Utilizes HuggingFace's "sentence-transformers/all-MiniLM-L6-v2" model to embed the text chunks.

- **Vector Store**

  Uses FAISS to build or load a local vector index for similarity search, keyed by an MD5 hash of the uploaded PDF file contents.

- **Retriever & QA Chain**

  The retriever fetches top-k relevant chunks, then passes them along with the user question to the LLM (Google Flan-T5 via HuggingFace pipeline) wrapped in LangChain's RetrievalQA.

- **User Interface**

Streamlit displays the upload widget, input box for questions, and the model's answers.

---

5. How to Use

1. Upload a PDF file containing your company's refund policy.
2. Wait for the text extraction and vector store indexing (cached for repeated uploads).
3. Enter a question about the refund policy in the text input box.
4. Receive an answer based on the uploaded document.
5. Repeat with new questions or upload another document.

---

6. Sample Inputs and Outputs

Example 1

Uploaded file: Refund_Policy_of_ipu.pdf (0.8MB)
User question:
What happens if a candidate applies for withdrawal after the last date of admission in medical courses?

Answer:

he/she can apply for withdrawal within the specified date, an amount of Rs. 1000/ - shall be deducted and the balance amount shall be refunded

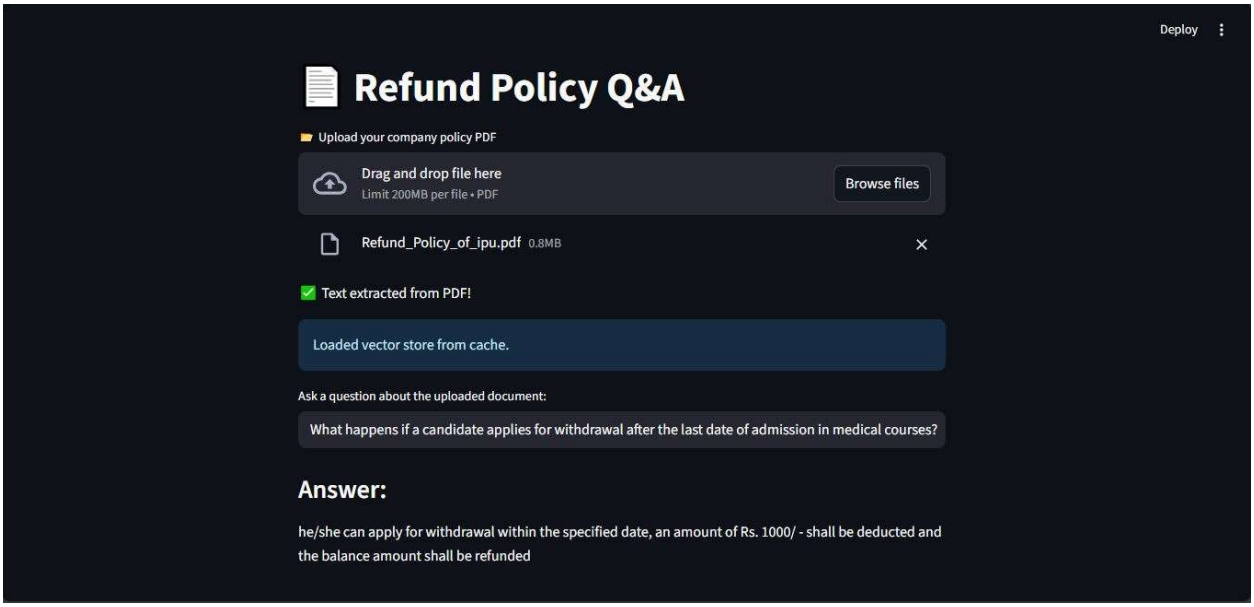


Fig1: Example1 Snapshot

Example 2

Uploaded file: Refund_Policy_of_ifinltd.pdf (61.4KB)
User question:
What are additional non-returnable items?

Answer:

Gift cards • Downloadable software products • Some health and personal care items

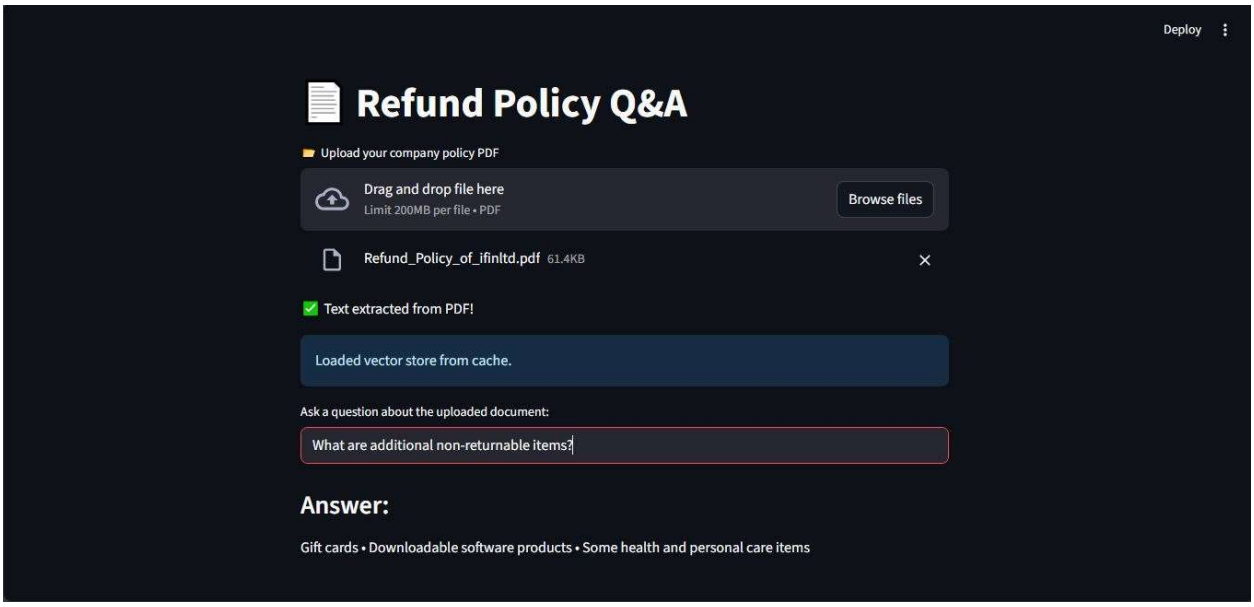


Fig2: Example2 Snapshot

---

7. Notes & Improvements

- **Caching:** The vector store is cached based on an MD5 hash of the uploaded file bytes, improving speed on repeated uploads of the same document.
- **Answer Quality:** Answers depend on the quality and scope of the uploaded PDF and the LLM used.
- **Extensions:**
    - Add multi-document upload and search.
    - Improve chunking strategy.
    - Experiment with larger or more powerful LLMs.
    - Highlight source text snippets in the answer.

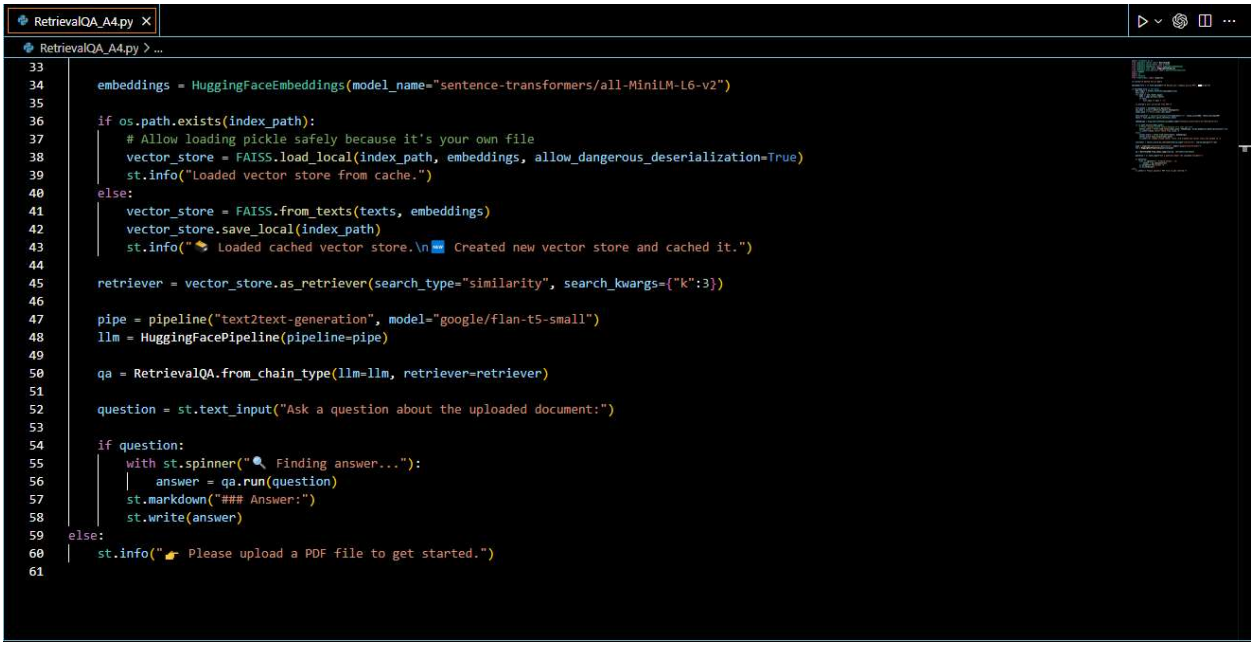
- Add logging and error handling for robustness.

---

8. Code Listing

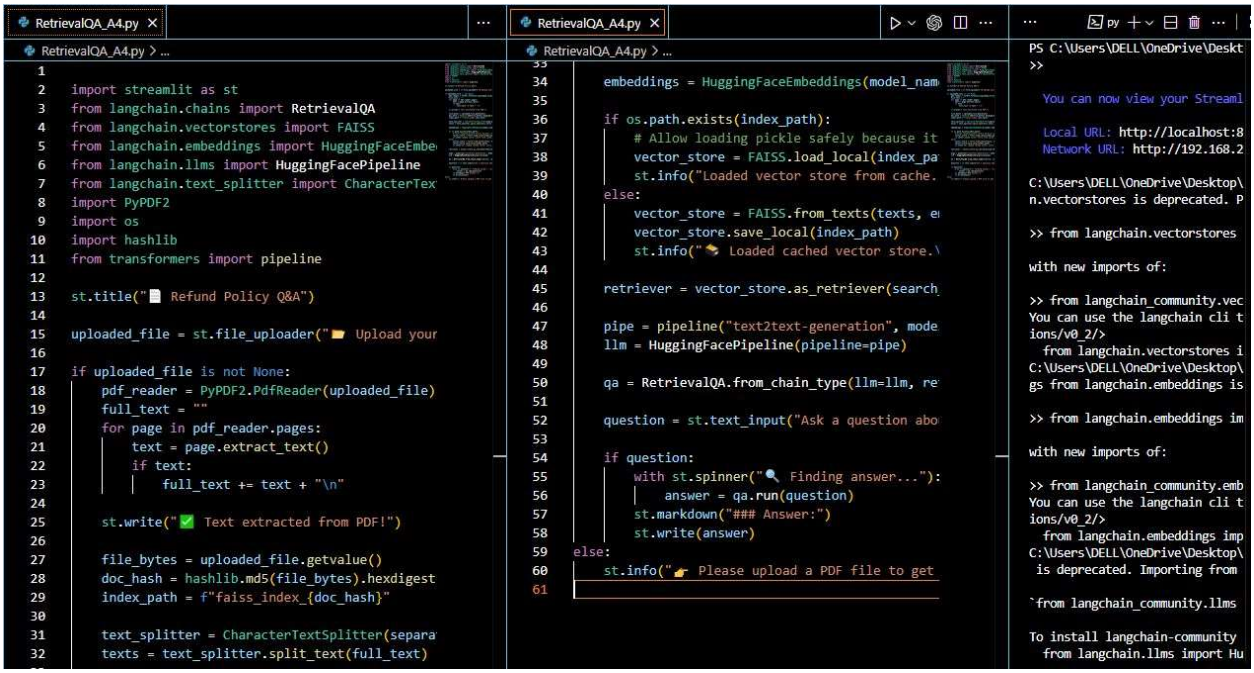
```python
import streamlit as st
from langchain.chains import RetrievalQA
from langchain.vectorstores import FAISS
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.llms import HuggingFacePipeline
from langchain.text_splitter import CharacterTextSplitter
import PyPDF2
import os
import hashlib
from transformers import pipeline

st.title("📄 Refund Policy Q&A")

uploaded_file = st.file_uploader("📂 Upload your company policy PDF", type=["pdf"])

if uploaded_file is not None:
    pdf_reader = PyPDF2.PdfReader(uploaded_file)
    full_text = ""
    for page in pdf_reader.pages:
        text = page.extract_text()
        if text:
            full_text += text + "\n"

    st.write("✅ Text extracted from PDF!")

    file_bytes = uploaded_file.getvalue()
    doc_hash = hashlib.md5(file_bytes).hexdigest()
    index_path = f"faiss_index_{doc_hash}"

    text_splitter = CharacterTextSplitter(separator="\n", chunk_size=1000, chunk_overlap=200)
    texts = text_splitter.split_text(full_text)
```

Fig3: Input Code Part1 Snapshot



```python
    embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")

    if os.path.exists(index_path):
        # Allow loading pickle safely because it's your own file
        vector_store = FAISS.load_local(index_path, embeddings, allow_dangerous_deserialization=True)
        st.info("Loaded vector store from cache.")
    else:
        vector_store = FAISS.from_texts(texts, embeddings)
        vector_store.save_local(index_path)
        st.info("🗃️ Loaded cached vector store.\n🆕 Created new vector store and cached it.")

    retriever = vector_store.as_retriever(search_type="similarity", search_kwargs={"k":3})

    pipe = pipeline("text2text-generation", model="google/flan-t5-small")
    llm = HuggingFacePipeline(pipeline=pipe)

    qa = RetrievalQA.from_chain_type(llm=llm, retriever=retriever)

    question = st.text_input("Ask a question about the uploaded document:")

    if question:
        with st.spinner("🔍 Finding answer..."):
            answer = qa.run(question)
        st.markdown("### Answer:")
        st.write(answer)
else:
    st.info("👉 Please upload a PDF file to get started.")
```

Fig4: Input Code Part2 Snapshot



```
RetrievalQA_A4.py
RetrievalQA_A4.py > ...

import streamlit as st
from langchain.chains import RetrievalQA
from langchain.vectorstores import FAISS
from langchain.embeddings import HuggingFaceEmbe
from langchain.llms import HuggingFacePipeline
from langchain.text_splitter import CharacterTex
import PyPDF2
import os
import hashlib
from transformers import pipeline

st.title("📄 Refund Policy Q&A")

uploaded_file = st.file_uploader("📂 Upload your

if uploaded_file is not None:
    pdf_reader = PyPDF2.PdfReader(uploaded_file)
    full_text = ""
    for page in pdf_reader.pages:
        text = page.extract_text()
        if text:
            full_text += text + "\n"

    st.write("✅ Text extracted from PDF!")

    file_bytes = uploaded_file.getvalue()
    doc_hash = hashlib.md5(file_bytes).hexdigest
    index_path = f"faiss_index_{doc_hash}"

    text_splitter = CharacterTextSplitter(separa
    texts = text_splitter.split_text(full_text)
```

```
RetrievalQA_A4.py
RetrievalQA_A4.py > ...

    embeddings = HuggingFaceEmbeddings(model_nam

    if os.path.exists(index_path):
        # Allow loading pickle safely because it
        vector_store = FAISS.load_local(index_pa
        st.info("Loaded vector store from cache.
    else:
        vector_store = FAISS.from_texts(texts, e
        vector_store.save_local(index_path)
        st.info("📚 Loaded cached vector store.\

    retriever = vector_store.as_retriever(search

    pipe = pipeline("text2text-generation", mode
    llm = HuggingFacePipeline(pipeline=pipe)

    qa = RetrievalQA.from_chain_type(llm=llm, re

    question = st.text_input("Ask a question abo

    if question:
        with st.spinner("🔍 Finding answer..."):
            answer = qa.run(question)
        st.markdown("### Answer:")
        st.write(answer)
else:
    st.info("👉 Please upload a PDF file to get
```

```
PS C:\Users\DELL\OneDrive\Deskt
>>

  You can now view your Streaml

  Local URL: http://localhost:8
  Network URL: http://192.168.2

C:\Users\DELL\OneDrive\Desktop\
n.vectorstores is deprecated. P

>> from langchain.vectorstores

with new imports of:

>> from langchain_community.vec
You can use the langchain cli t
ions/v0_2/>
  from langchain.vectorstores i
C:\Users\DELL\OneDrive\Desktop\
gs from langchain.embeddings is

>> from langchain.embeddings im

with new imports of:

>> from langchain_community.emb
You can use the langchain cli t
ions/v0_2/>
  from langchain.embeddings imp
C:\Users\DELL\OneDrive\Desktop\
 is deprecated. Importing from

`from langchain_community.llms

To install langchain-community
  from langchain.llms import Hu
```

Fig5: Input Code Snapshot