Introduction:

This Streamlit app provides a convenient and efficient way to extract business information from bank documents. It features a login authentication page to ensure the security of your data, and allows you to upload PDFs for processing. The app uses a large language model (LLM) to process the PDFs into chunks and word embeddings, and to extract information from images using optical character recognition (OCR).

Once the PDFs have been processed, you can ask the app queries to extract relevant information. For example, you could ask the app to identify all of the customer names in a PDF, or to extract the total amount of a loan. The app will then display the relevant information to you.

This app is designed to be used by businesses of all sizes to extract business information from bank documents. It is easy to use and does not require any coding knowledge. The app is also highly accurate and reliable, making it the perfect tool for businesses that need to extract information from large volumes of bank documents.

Benefits

- Easy to use: The app is easy to use, even for users with no coding knowledge.

- Secure: The app features a login authentication page to ensure the security of your data.

Use Cases:

The app can be used to extract business information from a variety of bank documents, including:

- Account statements

- Loan agreements

- Credit card statements

- Insurance policies

Workflow:
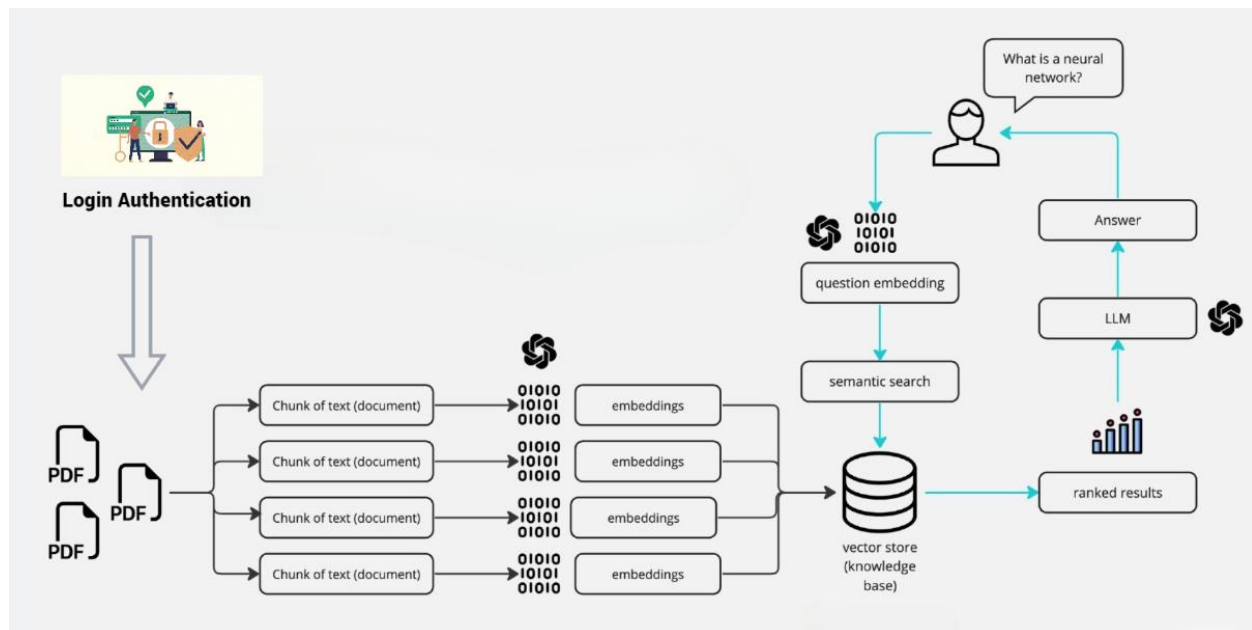
Steps :

1. Start the App.

2. User authenticates by giving proper username and password.

3. User uploads a PDF or image file.

4. If the file is a PDF, convert it to text. If the pdf contains images, use OCR to convert the images to texts.

5. Convert the texts into many smaller chunks of 1000 words each.

6. Generate word embedding of the chunks using LLM.

7. Store the word embedding into a vector database (FAISS).

8. User asks a question about the information in the PDF.

9. Generate question embedding.

10. Perform semantic search of the "question embedding" to find the most similar embeddings stored in the vector database and rank them according to the similarity score.

11. Pass the ranked results and question to the LLM.

12. The previous chat history is also passed to the LLM.

13. Analyzing the information LLM answers the query.
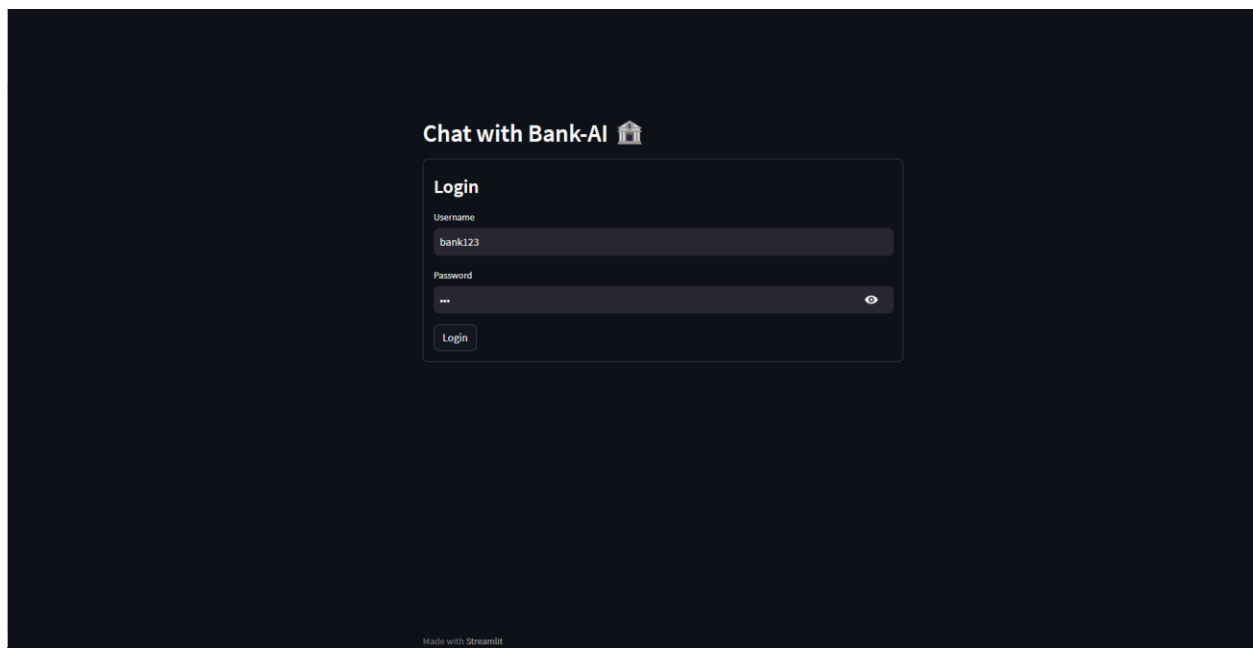
14. Showing the answer to the user.

15. End

Framework and Library Used :

- Streamlit for UI and backend.
- PyPDF2 and fitz for reading texts from PDF.
- Pytesseract for performing OCR.
- FAISS for storing the word embedding of the texts and find the most similar embedding based on semantic search.
- Langchain for loading LLM storing memory of the Chat.
- HuggingFaceHub for accessing LLM.
- Streamlit_authentication is Used for authentication.

LLM used :

- "hkunlp/instructor-large" from HuggingFaceHub is used to generate word embedding of the text chunks of the loaded PDF.
- "google/flan-t5-large" is used to answer the query of the user and understand the context of chat. Parameter tuned : "temperature" :0.5,"max_length" :1024.

Login Page:

Main Page:

Upload PDF and press process :

Ask question and get Answer:

Your Documents

Upload your PDFS here and click on 'Process'

Drag and drop file here
Limit 200MB per file

Browse files

Chapter 4 - Sources of Fina... ✕
122.5KB

Process

Chat with Bank-AI 🏛️

Ask a question about uploaded documents...

write down the long term sources of finances

Submit

write down the long term sources of finances

Equity Shares Preference Shares Debenture Long-term Loans Fixed Deposits