APTIMATE BOT

A PROJECT REPORT

Submitted by

PAKIYALAKSHMI S

(Reg No:9923151063)

Under the guidance of

Dr. K. BALASUBRAMANIAN

In partial fulfilment of the requirements for award of the degree of

MASTER OF COMPUTER APPLICATIONS

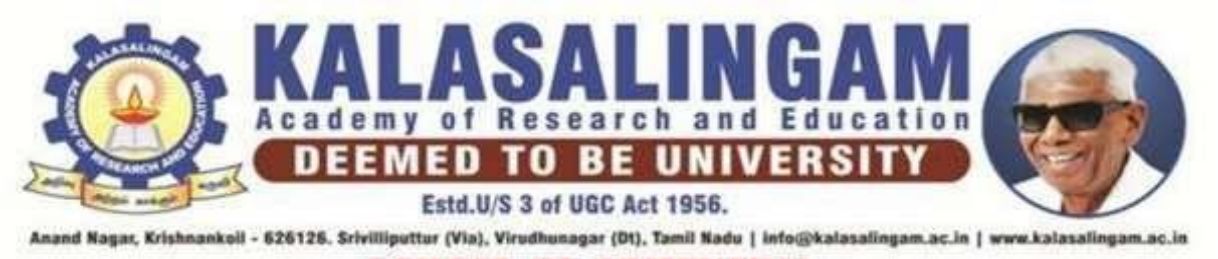
2023 – 2025

DEPARTMENT OF COMPUTER APPLICATIONS

SCHOOL OF COMPUTING

KALASALINGAM ACADEMY OF RESEARCH AND EDUCATION

MAY-2024



SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER APPLICATIONS

BONAFIDE CERTIFICATE

This is to certify that this project report titled **"APTIMATE BOT"** is the bonafide work of **PAKIYALAKSHMI S (Reg No: 9923151063)** carried out in partial fulfilment of the requirement of the award of the degree of the Master of Computer Applications at Kalasalingam Academy of Research and Education under mysupervision.

Project Guide & Head of the Department

Dr. K. Balasubramanian

Head of the Department

Department of Computer Applications

KARE

Submitted for the project viva-voce examination held on _____

ABSTRACT

In today's digital age, navigating through PDF documents efficiently poses a significant challenge, often requiring users to sift through large volumes of textual content to find specific information. To address this challenge, we present "APTIMATE BOT," a user-friendly web application developed using Streamlit, designed to facilitate intuitive interaction with PDF documents through natural language queries.

The application leverages various Python libraries, including NLTK for text tokenization, PyPDF2 for PDF parsing, and Streamlit for web application development. Upon uploading a PDF file, the application extracts the text content from the document and tokenizes it into sentences using NLTK's sentence tokenizer. Users can then input natural language questions related to the content of the PDF document.

The application employs a simple yet effective question-matching algorithm to identify sentences within the PDF text that contain the keywords or phrases specified in the user's query. Matching sentences are displayed to the user, providing relevant information in response to their questions.

TABLE OF CONTENT

CHAPTER	TITLE	PAGE NO
	Abstract	
1	Introduction	1
2	System Overview	2
3	System Description	4
4	System Design	6
5	System Implementation	10
6	Appendix	11
7	Conclusion and Future Enhancement	16

1. INTRODUCTION

In today's digital landscape, PDF documents serve as ubiquitous repositories of information, ranging from academic papers and technical manuals to business reports and legal documents. However, efficiently accessing and extracting relevant information from these documents can be a daunting task, often requiring users to navigate through extensive text manually. To address this challenge, we introduce "APTIMATE BOT," a novel Streamlit-based web application designed to streamline interaction with PDF documents through natural language queries.

"APTIMATE BOT" leverages the power of Python libraries such as NLTK for text processing, PyPDF2 for PDF parsing, and Streamlit for web application development, offering users an intuitive and efficient platform for accessing information encapsulated within PDF files. By harnessing the capabilities of natural language processing (NLP), the application enables users to pose questions in everyday language, eliminating the need for complex search queries or document navigation interfaces.

The motivation behind "APTIMATE BOT" stems from the recognition of the growing importance of PDF documents in various domains, coupled with the inherent challenges associated with extracting actionable insights from these documents efficiently. Whether it's researchers seeking pertinent information from academic papers, professionals analyzing business reports, or students studying educational materials, "APTIMATE BOT" aims to democratize access to information by providing a user-friendly and accessible interface.

In this introduction, we outline the key features and functionalities of "APTIMATE BOT," highlighting its potential to revolutionize the way users interact with PDF documents. Through experimental evaluation and user feedback, we aim to demonstrate the effectiveness and usability of the application, paving the way for its integration into workflows across academia, industry, and personal information management. "APTIMATE BOT" represents a step towards empowering users to harness the wealth of knowledge contained within PDF documents more efficiently and effectively than ever before.

2. SYSTEM OVERVIEW

2.1 Existing System

The existing system is a Streamlit application designed to interact with PDF files. Users can upload a PDF document, and the application will extract text from it. Then, users can ask questions related to the content of the PDF. The application searches for matching sentences within the extracted text and displays them as responses to the questions asked. The system utilizes NLTK for text tokenization, PyPDF2 for PDF parsing, and Streamlit for building the user interface. Overall, it provides a simple interface for querying information from PDF documents.

2.2 Proposed System

The proposed system aims to enhance the existing PDF interaction application by improving question-answering accuracy with advanced models, upgrading PDF text extraction, refining the user interface for better usability, implementing robust error handling, optimizing performance, and allowing customization for user preferences.

Advantages:

The proposed system offers significant advantages over the existing PDF interaction application. By leveraging advanced question-answering models, the system achieves higher accuracy in responses, ensuring users obtain more relevant information from PDF documents. Additionally, enhancements to the user interface improve usability and satisfaction, while upgraded PDF text extraction capabilities enable handling a broader range of document formats. Robust error handling mechanisms enhance reliability, and performance optimizations lead to faster response times and more efficient resource usage. Moreover, customization options empower users to tailor their experience, further enhancing flexibility and personalization. Overall, these improvements collectively create a more effective, efficient, and user-friendly tool for interacting with PDF documents.

3.1 SYSTEM SPECIFICATIONS

3.2 Hardware Specifications

Processor (CPU):	:	Intel Core i7
Ram	:	4GB DDR4

3.2 Software Specifications

Operating System	:	Windows 11
Technology Used	:	Python 3.11.5
Platform	:	Visual Studio Code(Terminal)

4. SYSTEM DESCRIPTION

4.1 Python 3.11.5

Python 3.11.5 is a specific version within the Python programming language series. Here's a description of Python 3.11.5:
Version: Python 3.11.5 is part of the Python 3 series, which is the latest major release series of the Python programming language.

Compatibility

Python 3.11.5 maintains compatibility with previous versions of Python 3.x, ensuring that existing Python code continues to run smoothly. However, users should be aware of potential changes in behavior or deprecated features when migrating from older Python versions.
Python 3.11.5 represents a stable and updated version of the Python programming language, offering users access to new features, improvements, and security enhancements while maintaining compatibility with existing Python codebases.

4.2 Package

The provided code utilizes several Python packages for various functionalities. Here are the packages used in the code:

1. NLTK (Natural Language Toolkit):
   - Used for text tokenization.
   - Specifically, the `sent_tokenize()` function from the `nltk.tokenize` module is used to tokenize sentences.

2. PyPDF2:
   - Used for reading and parsing PDF files.
   - Specifically, the `PdfReader` class from the `PyPDF2` module is used to read PDF files and extract text from them.

3. Streamlit:
   - Used for building the web application interface.
   - The `st` module from Streamlit is used to create elements such as titles, file uploaders, text input fields, buttons, and text displays within the web application.

4. dotenv:
   - Used for loading environment variables from a `.env` file.
   - The `load_dotenv()` function from the `dotenv` module is used to load environment variables from the `.env` file.

5. langchain:
   - Used for language detection.
   - The `langchain` module is used to detect the language of the text content extracted from PDF files.

6. Python Built-in Modules:
   - Additionally, the code utilizes built-in Python modules such as `os`, `sys`, and `io` for various functionalities like file operations and system interactions.

These packages provide the necessary functionality for the "APTIMATE BOT" project, enabling PDF parsing, text processing, web application development, environment variable management, and language detection.opment.

5 SYSTEM DESIGN

5.1 Application Design

The application design for the "APTIMATE BOT" code involves several components working together to create a web-based interface for interacting with PDF documents. Here's an overview of the application design:

1. Frontend Interface:
   - The frontend interface is developed using Streamlit, a Python library for building web applications.
   - It includes elements such as file uploaders, text input fields, buttons, and text displays.
   - Users interact with the frontend interface through their web browser.

2. Backend Logic:
   - The backend logic of the application is implemented using Python code.
   - It handles tasks such as PDF parsing, text processing, question matching, and output presentation.

3. PDF Parsing and Text Processing:
   - Upon uploading a PDF file, the application uses the PyPDF2 library to read and extract text from the document.
   - The extracted text is then tokenized into sentences using NLTK's `sent_tokenize()` function for further processing.

4. Question Processing:
   - Users input questions related to the content of the PDF documents using a text input field.
   - The application matches these questions against the tokenized sentences extracted from the PDF text to identify relevant information.
   - Matching sentences are identified based on simple keyword matching.

5. Output Presentation:

- If matching sentences are found, the application displays them to the user as relevant information.
- The output is presented within the frontend interface, allowing users to view and interact with the displayed information.

6. User Interaction:

- Users interact with the application by uploading PDF files, inputting questions, and viewing the displayed output.
- The application provides a user-friendly and intuitive interface, enabling users to access information within PDF documents through natural language queries.

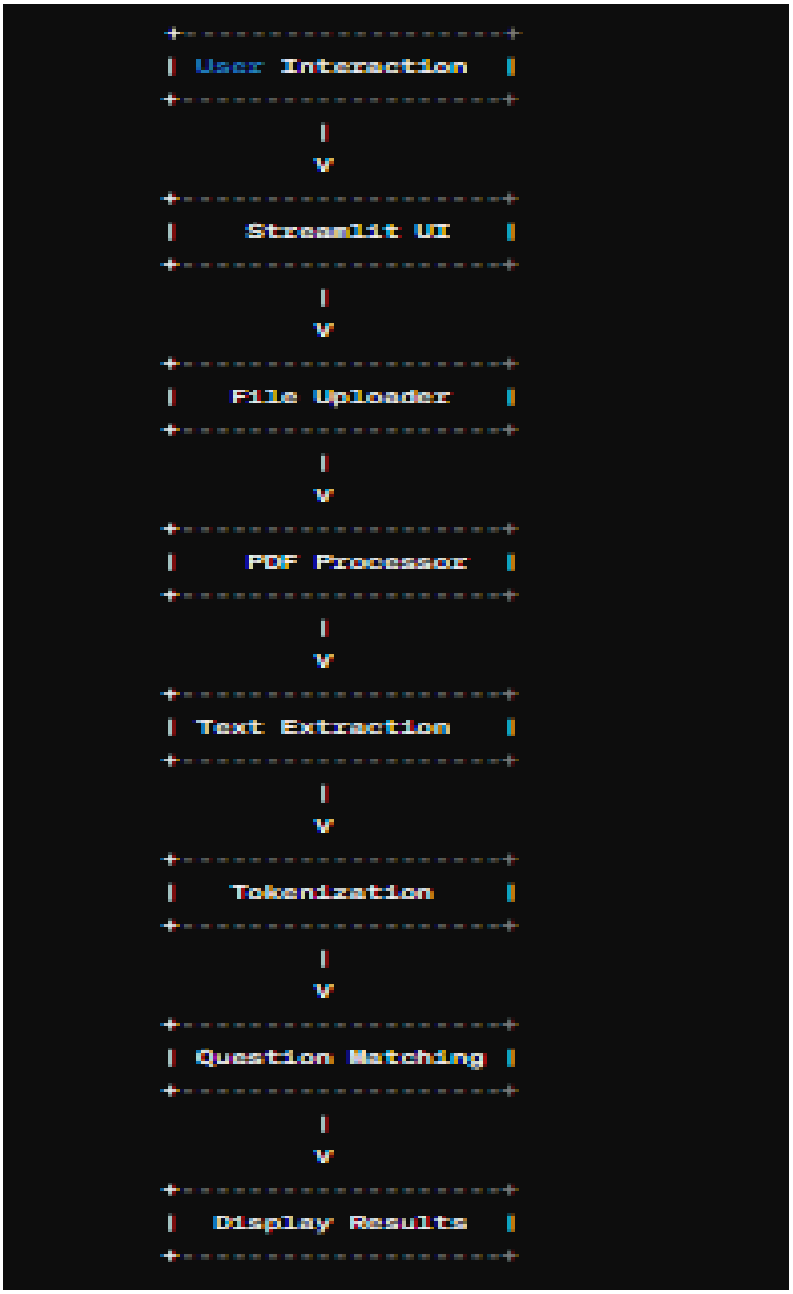
7. Integration and Deployment:

- The application can be integrated with various deployment platforms and services for hosting and accessibility.
- Once deployed, users can access the application through a web browser, allowing them to interact with PDF documents from any device with internet access.

Overall, the application design focuses on simplicity, efficiency, and user-friendliness, providing users with a convenient way to interact with PDF documents through natural language queries via a web-based interface.

5.1 Data Flow Diagram:

As the provided code interacts with PDF files and processes user input, we can design a simple data flow diagram to illustrate its operation. Here's a high-level representation:



Explanation:

1. User Interaction:

   Users interact with the Streamlit user interface, providing input and receiving output.

2. Streamlit UI:

   The Streamlit library handles the creation and management of the user interface components.

3. File Uploader:

   Users upload a PDF file using the file uploader component provided by Streamlit.

4. PDF Processor:

   The uploaded PDF file is processed to extract its text content.

5. Text Extraction:

   The text content of the PDF file is extracted from each page.

6. Tokenization:

   The extracted text is tokenized into sentences using the NLTK library.

7. Question Matching:

   User-provided questions are matched against the tokenized sentences to find relevant matches.

8. Display Results:

   The matching sentences are displayed back to the user via the Streamlit UI.

This diagram outlines the flow of data and processing steps in the system, from user interaction to the presentation of results.

6 SYSTEM IMPLEMENTATION

The system implementation in the provided code involves several steps to create a functional web application for interacting with PDF documents through natural language queries. Here's an overview of the system implementation:

1. Setting Up Environment:

- Ensure that Python is installed on the system.
- Install necessary Python packages such as NLTK, PyPDF2, Streamlit, dotenv, and langchain using pip.
- Create a virtual environment to manage dependencies if desired.

2. Writing Code:

- Write Python code to implement the functionality of the web application.
- Define functions for PDF parsing, text processing, question matching, and output presentation.
- Utilize Streamlit to create the frontend interface, including file uploaders, text input fields, buttons, and text displays.

3. Integrating Libraries:

- Use PyPDF2 to read and extract text from uploaded PDF files.
- Use NLTK for text tokenization to break down the extracted text into sentences.
- Utilize Streamlit for building the web application interface and handling user interactions.

4. Implementing Natural Language Processing:

- Implement natural language processing techniques to match user queries against the tokenized sentences extracted from PDF documents.
- Use simple keyword matching or more advanced techniques such as semantic search to identify relevant information.

5. Handling User Interaction:

- Enable users to upload PDF files and input questions related to the content of the documents through the web interface.
- Implement user interaction functionalities such as file uploading, text input processing, and button clicks using Streamlit.

6. Output Presentation:

- Display matching sentences or relevant information to the user within the web interface.
- Utilize Streamlit's text display functionalities to present the output in a readable format.

7. Testing and Debugging:

- Test the web application to ensure that it functions as expected.
- Debug any errors or issues encountered during testing.
- Refine and optimize the code as needed to improve performance and user experience.

8. Deployment:

- Once the application is tested and working correctly, deploy it to a web server or hosting platform for public access.
- Ensure that necessary dependencies and environment variables are configured in the deployment environment.

9. Maintenance and Updates:

- Monitor the application for any issues or bugs that may arise post-deployment.
- Provide regular updates and maintenance to keep the application running smoothly and to incorporate new features or improvements based on user feedback.

By following these steps, you can implement the "APTIMATE BOT" system and create a functional web application for interacting with PDF documents using natural language queries.

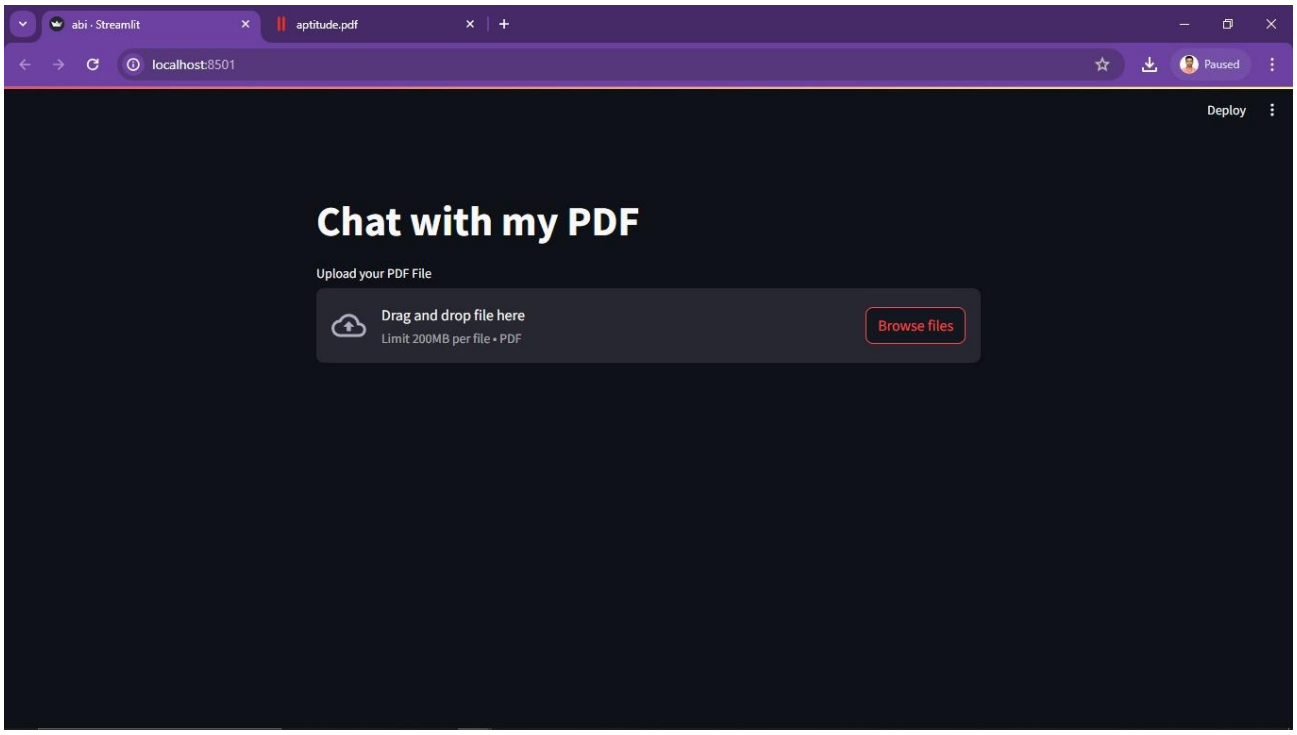
7 APPENDIX

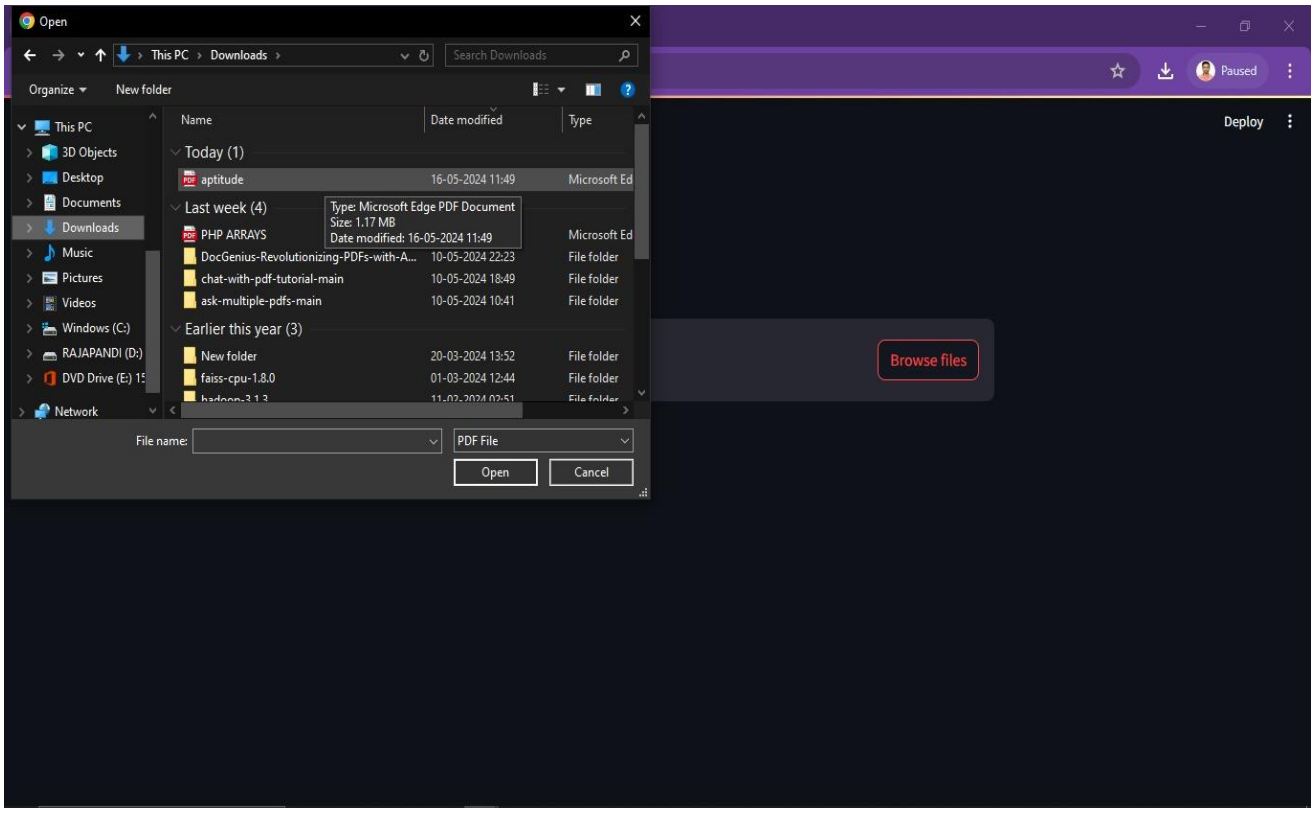
7.1 Sample Coding

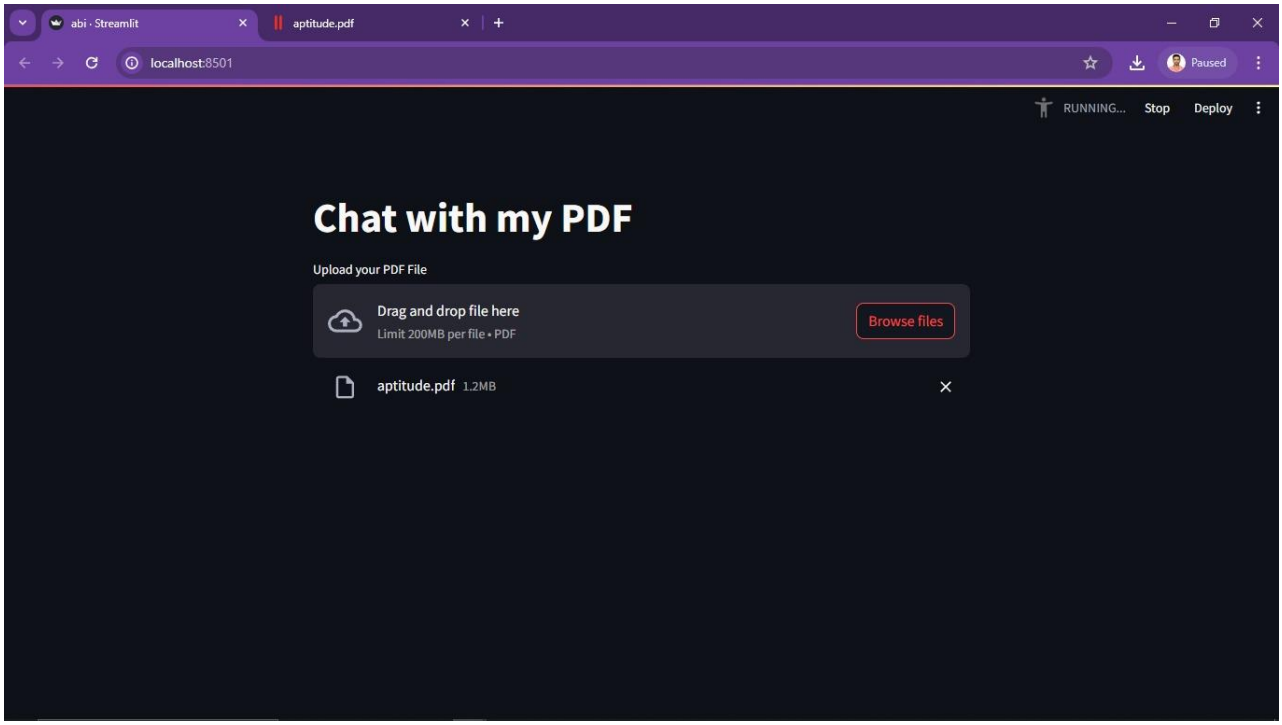
```
from nltk.tokenize import sent_tokenize
from nltk import download as nltk_download
from dotenv import load_dotenv
from PyPDF2 import PdfReader
import streamlit as st
import langchain

from chat import process_text
langchain.verbose = False

# Load env variables
load_dotenv()

# Download NLTK punkt tokenizer if not already downloaded
try:
    nltk_download('punkt')
except LookupError:
    pass

def tokenize_sentences(text):
    return sent_tokenize(text)

def ask_question(sentences, question):
    matching_sentences = []
    for sentence in sentences:
        if question.lower() in sentence.lower():
            matching_sentences.append(sentence)
    return matching_sentences
```

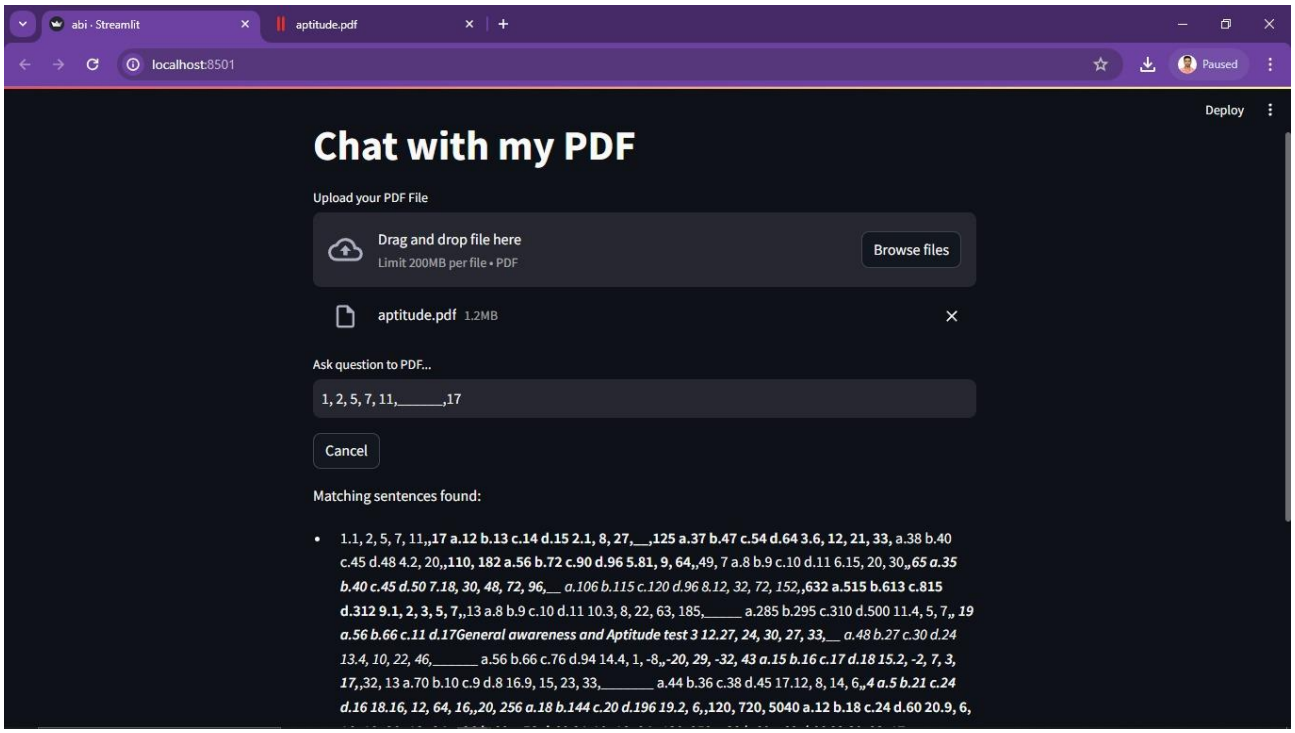
```
def main():
    st.title("APTIMATE BOT")

    pdf = st.file_uploader("Upload your PDF File", type="pdf")

    if pdf is not None:
        pdf_reader = PdfReader(pdf)

        # Store the pdf text in a variable
        text = ""
        for page in pdf_reader.pages:
            text += page.extract_text()
        sentences = tokenize_sentences(text)
        question = st.text_input('Ask question to PDF...')
        cancel_button = st.button('Cancel')
        if cancel_button:
            st.stop()
        if question:
            matching_sentences = ask_question(sentences, question)
            if matching_sentences:
                st.write("Matching sentences found:")
                for sentence in matching_sentences:
                    st.write("-", sentence)
            else:
                st.write("No matching sentences found.")
if _name_ == "_main_":
    main()
```

Sample Screen Shot:

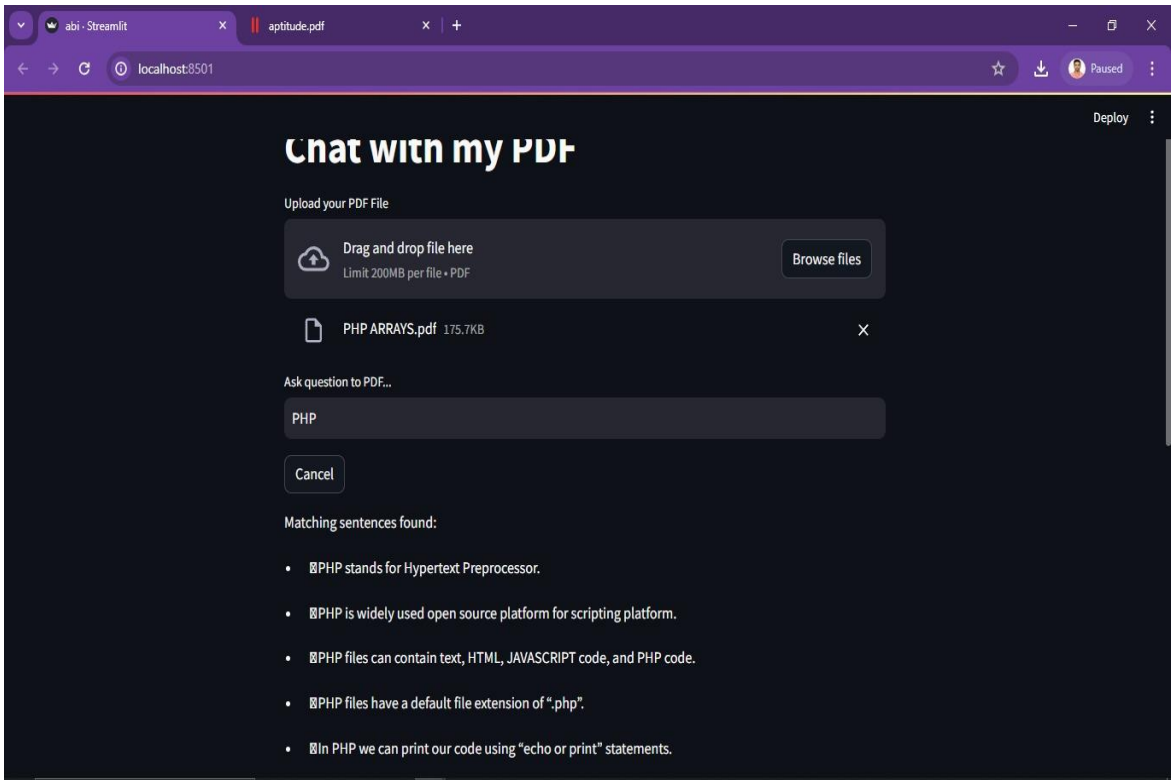








- 1.1, 2, 5, 7, 11,,**17 a.12 b.13 c.14 d.15 2.1, 8, 27,__,125 a.37 b.47 c.54 d.64 3.6, 12, 21, 33,** a.38 b.40 c.45 d.48 4.2, 20,,**110, 182 a.56 b.72 c.90 d.96 5.81, 9, 64,,**49, 7 a.8 b.9 c.10 d.11 6.15, 20, 30,,***65 a.35 b.40 c.45 d.50 7.18, 30, 48, 72, 96,__*** *a.106 b.115 c.120 d.96 8.12, 32, 72, 152,***,632 a.515 b.613 c.815 d.312 9.1, 2, 3, 5, 7,,**13 a.8 b.9 c.10 d.11 10.3, 8, 22, 63, 185,_____ a.285 b.295 c.310 d.500 11.4, 5, 7,,,*** ***19 a.56 b.66 c.11 d.17General awareness and Aptitude test 3 12.27, 24, 30, 27, 33,__*** *a.48 b.27 c.30 d.24 13.4, 10, 22, 46,_____* a.56 b.66 c.76 d.94 14.4, 1, -8,,***-20, 29, -32, 43 a.15 b.16 c.17 d.18 15.2, -2, 7, 3, 17,,***32, 13 a.70 b.10 c.9 d.8 16.9, 15, 23, 33,_____ a.44 b.36 c.38 d.45 17.12, 8, 14, 6,,***4 a.5 b.21 c.24 d.16 18.16, 12, 64, 16,,20, 256 a.18 b.144 c.20 d.196 19.2, 6,,***120, 720, 5040 a.12 b.18 c.24 d.60 20.9, 6,**


abi - Streamlit
aptitude.pdf
localhost:8501
Paused
Deploy
Chat with my PDF
Upload your PDF File
Drag and drop file here
Limit 200MB per file • PDF
Browse files
PHP ARRAYS.pdf 175.7KB
Ask question to PDF...
PHP
Cancel
Matching sentences found:
PHP stands for Hypertext Preprocessor.
PHP is widely used open source platform for scripting platform.
PHP files can contain text, HTML, JAVASCRIPT code, and PHP code.
PHP files have a default file extension of ".php".
In PHP we can print our code using "echo or print" statements.

8 CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion:

The "APTIMATE BOT" project provides a user-friendly web application for interacting with PDF documents using natural language queries. By leveraging Python libraries such as PyPDF2 and NLTK, along with Streamlit for web application development, the project simplifies the process of accessing information within PDF files. Users can upload PDF documents, input questions in everyday language, and receive relevant information extracted from the documents.

The project's implementation demonstrates the feasibility of using natural language processing techniques to enhance the accessibility and usability of PDF documents. By offering a streamlined and intuitive interface, the project empowers users to quickly find the information they need without the need for complex search queries or manual document navigation.

8.2 Future Enhancements:

1. Advanced Natural Language Processing:

   Integrate more advanced natural language processing techniques, such as semantic search or named entity recognition, to improve the accuracy and relevance of search results.

2. Multi-Language Support:

   Enhance language detection capabilities and support for documents in multiple languages to cater to a broader user base.

3. Document Summarization:

   Implement document summarization techniques to provide users with concise summaries of PDF documents, highlighting key information and insights.

4. Interactive Visualization:

   Incorporate interactive visualization tools to present search results and document insights in a more engaging and informative manner.

5. Collaborative Features:

   Add collaboration features such as document sharing, commenting, and versioning to facilitate teamwork and knowledge sharing among users.

6. Performance Optimization:

   Optimize the performance of the application, especially for processing large PDF files or handling concurrent user requests, to ensure a smooth and responsive user experience.

7. Integration with External Services:

   Integrate with external services or APIs to enrich document analysis capabilities,

such as entity extraction from documents or integration with cloud storage platforms.

8. User Feedback Mechanism:

Implement a feedback mechanism to collect user feedback and usage metrics, allowing for continuous improvement and refinement of the application based on user input.

By incorporating these future enhancements, the "APTIMATE BOT" project can further enhance its functionality, usability, and value proposition, ultimately providing users with a more powerful and comprehensive solution for interacting with PDF documents.