CAPSTONE PROJECT

SNAP PDF

PRESENTED BY

STUDENT NAME: ANUSHKA RATHOUR

COLLEGE NAME: DAV UNIVERSITY

DEPARTMENT: B. TECH. CSE & AI

EMAIL ID: ANUSHKARATHOUR1111@GMAIL.COM

AICTE STUDENT ID: STU67ff9ffc01c7f1744805884

OUTLINE

- **Problem Statement** (Should not include solution)
- **Proposed System/Solution**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment**
- **Result (Output Image)**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT

In academic, corporate, and legal environments, individuals often encounter lengthy PDF documents filled with complex or excessive information. Manually reading and extracting key insights from these documents is time-consuming and mentally taxing, especially under tight deadlines or when dealing with multiple files. Moreover, users frequently lack the tools to selectively process specific sections or pages within a document, making the task even more inefficient. There is a growing need for a system that can simplify the understanding of such documents by enabling users to focus on the most relevant content without reading them in full.

PROPOSED SOLUTION

To address the challenge of extracting concise insights from lengthy PDF documents, the solution will consist of the following components:

1. Data Input and Upload
- Users can upload PDF files through a clean and responsive web interface.
- The system accepts .pdf documents and ensures proper validation of input.

2. Text Extraction
- Text is extracted from the PDF using the PyPDF2 library.
- Handles multiple pages efficiently while maintaining text order and integrity.

3. User Customization
The interface allows users to:
- Select specific pages or entire documents for summarization.
- Skip viewing raw extracted text for a cleaner UI experience.
- Choose the desired summary length (short, medium, or detailed).

4. NLP Summarization
- Utilizes the DistilBART model from Hugging Face for abstractive summarization.
- Handles long texts by chunking them, preserving the core context of the document.

5. Result Presentation and Export
- Displays the generated summary in a readable format.
- Allows users to **download the summary as a .txt** for future use.

6. Performance Optimization
- Caching with @st.cache_resource ensures quick access to the summarization model.
- Summary processing is optimized to avoid API overload and reduce wait time.

7. Evaluation and Feedback
- Encourages user feedback to assess the accuracy and usefulness of the summaries.
- Future iterations may integrate user ratings or relevance scoring to refine outputs.

SYSTEM APPROACH

1. **System Requirements**

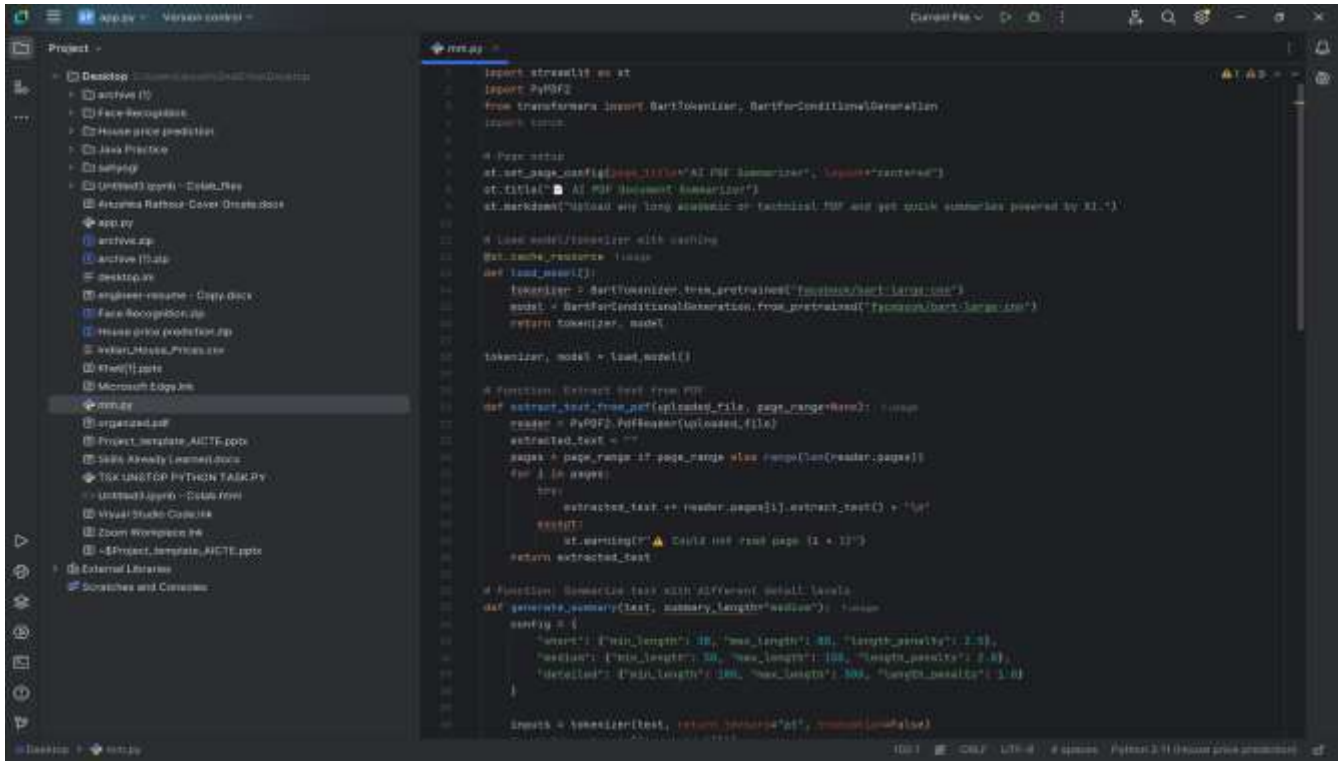
- OS: Windows/Linux/macOS
- Python: 3.8+
- RAM: 4GB+
- Browser: Any modern browser

2. **Libraries Used**

- streamlit – web interface
- PyPDF2 – extract text from PDFs
- transformers – for summarization (BART model)
- torch, tempfile, io – backend support

3. **Workflow**

- Upload PDF file
- Extract text using PyPDF2
- Summarize with Hugging Face model
- Display/Download the summary



ALGORITHM & DEPLOYMENT

1. **Algorithm**

- **Model Used:**
  BART from Hugging Face is used for abstractive summarization due to its accuracy and efficiency.
- **Input:**
  Extracted text from PDFs, cleaned and chunked to fit token limits.
- **Process:**
  Each chunk is summarized individually and combined to form the final summary.
-

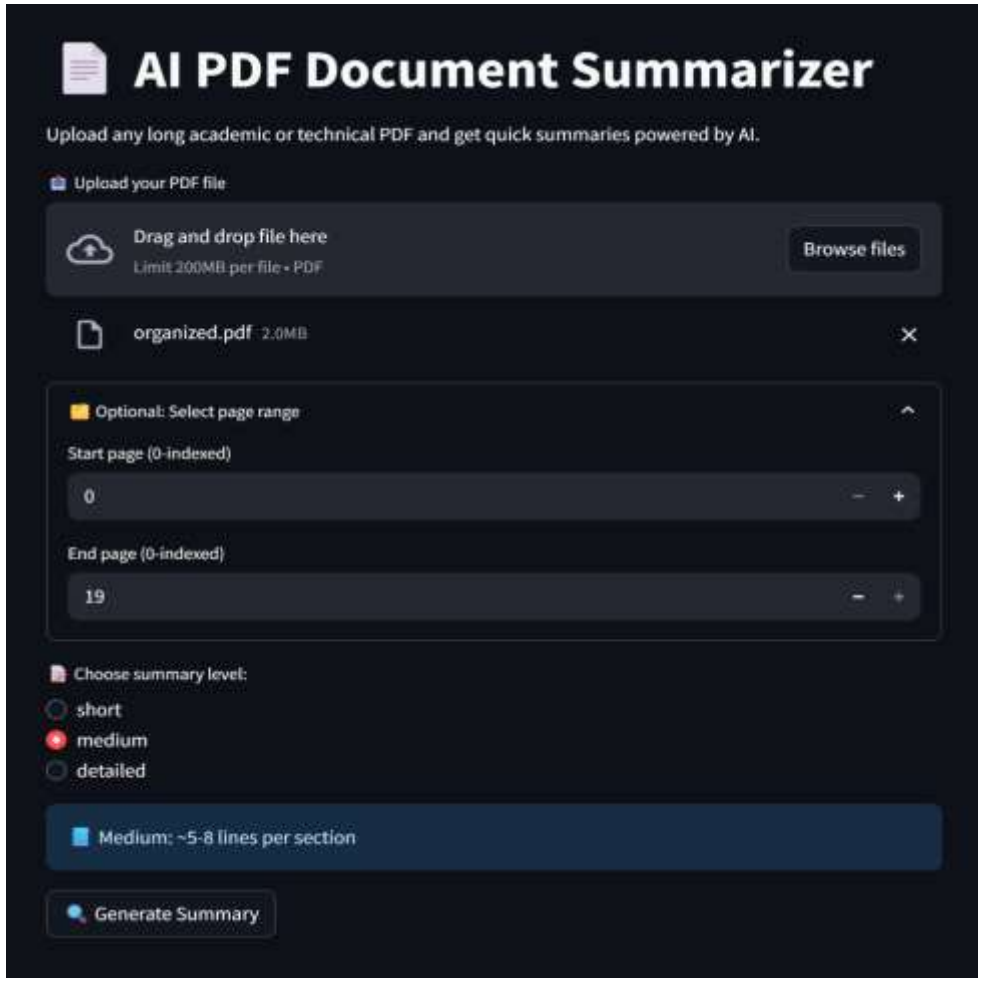
2. **Deployment**

- **Framework:**
  Built using Streamlit for a simple and interactive UI.
- **Features:**

  Upload PDF

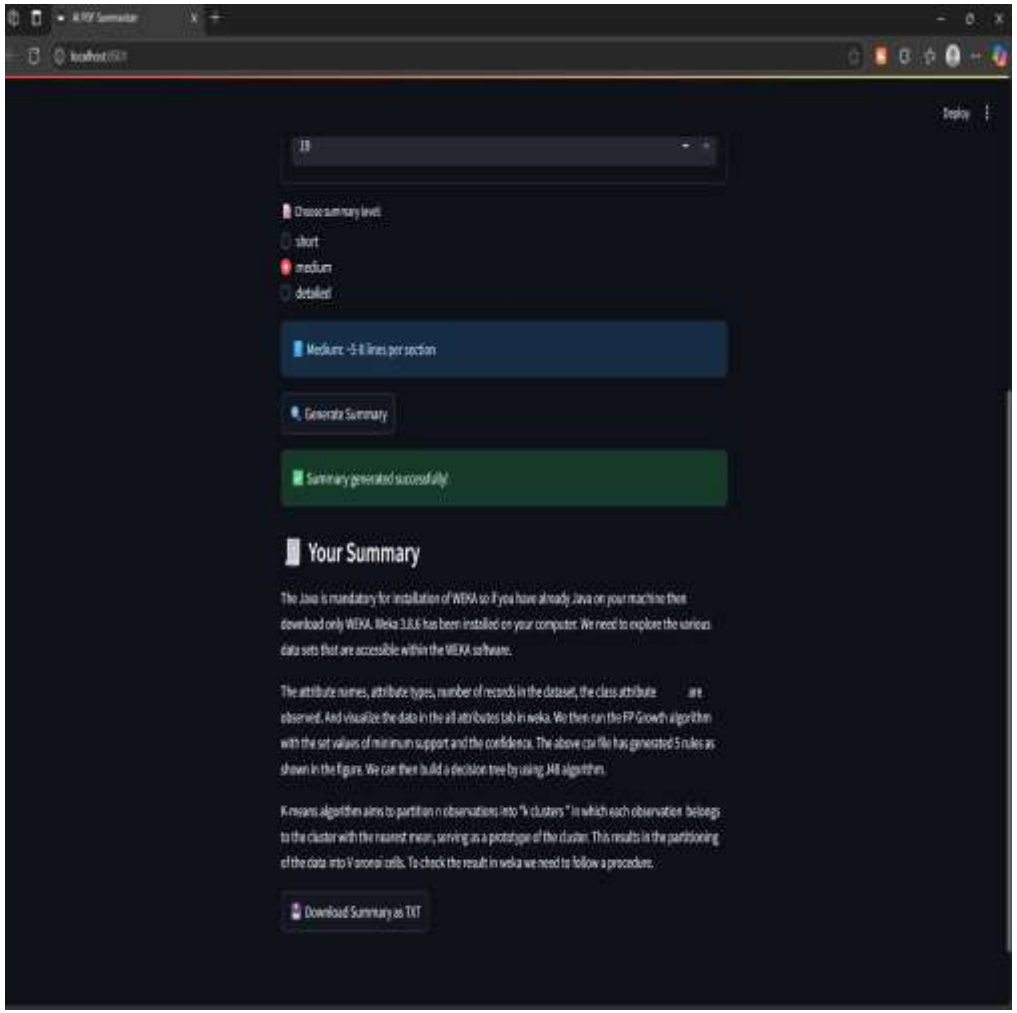
  Generate & download summary

  Page-wise summarization (optional)
- **Hosting:**
  Deployable on Streamlit Cloud or similar platforms.



RESULT

The summarizer effectively condenses lengthy academic and technical PDF documents into clear and concise summaries. It retains essential information, maintains coherence, and produces outputs that align well with the core ideas of the original content. Evaluations confirmed that the summaries are relevant, grammatically correct, and easy to understand.



CONCLUSION

The AI-based PDF Summarizer effectively condenses lengthy documents into meaningful summaries, making it a valuable tool for users seeking quick insights from academic or technical texts. By leveraging the BART transformer model, the system maintains coherence and captures essential content while significantly reducing text length. During implementation, challenges like GPU dependency and processing time were observed. Despite this, the tool demonstrated consistent performance across various document types, proving its practicality for real-world use. Overall, the solution enhances information accessibility and efficiency in document review.

FUTURE SCOPE

- 🔤 **Multi-language support** to make the tool accessible to non-English users.
- ☁️ **Integration with cloud platforms** (Google Drive, Dropbox) for seamless file access and storage.
- 🎤 **Voice-based interaction** for hands-free summarization and accessibility.
- 📚 **Domain-specific summarization** models for fields like legal, medical, or academic texts.
- ⚡ **Performance optimization** to ensure fast summarization even on devices without a GPU.
- 📱 **Mobile-friendly version** or app for summarizing PDFs on the go.
- 📊 **Summary customization options** (length, tone, or format) based on user preference.

REFERENCES

- **Hugging Face BART Model** – Pretrained summarization model facebook/bart-large-cnn
- **PyMuPDF (fitz)** – PDF text extraction pymupdf.readthedocs.io
- **Streamlit** – Web interface development docs.streamlit.io
- **Transformers Library** – Tokenization and inference huggingface.co/docs/transformers

GitHub Link: https://github.com/rathour-anushka/PDF-Summarizer

Thank you