A project on

Book PDF Downloader

Guided By :-

Ms. Aastha Sharma

Submitted By :-

Shubham, Priyansh, Ishaan, Adarsh

INDEX:-

INTRODUCTION

- **PDF Book Downloader Project**
- **Objective:** Enhancing digital access to educational materials through a user-friendly application.
- **Methodology:** Developed using Python, Tkinter, requests library for web scraping, and BeautifulSoup for HTML parsing.
- **Key Features:** Search functionality across multiple engines, PDF retrieval, and robust download management.
- **Importance:** Facilitating convenient access to educational resources globally.

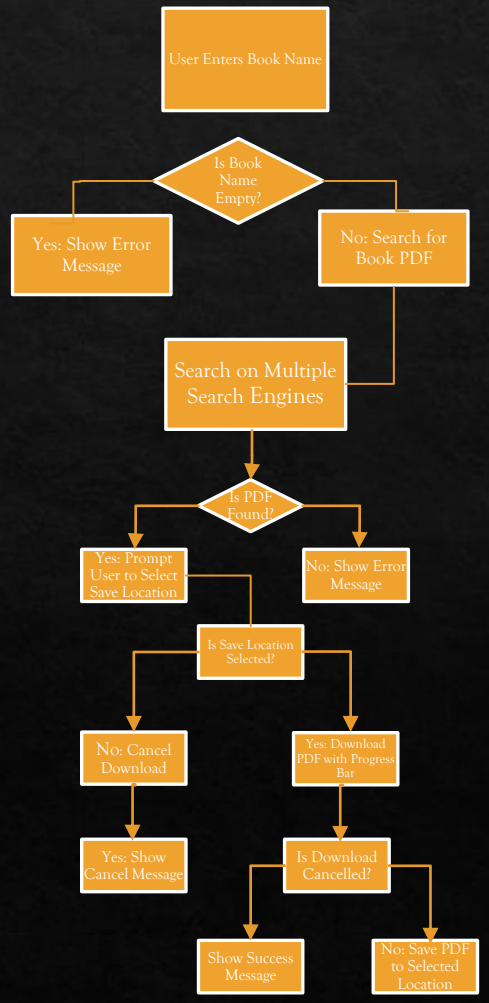
OBJECTIVES

- **Develop a User-Friendly Interface:** Create an intuitive interface for users to search and download PDF books effortlessly.
- **Implement Search Functionality:** Enable searching across multiple search engines to locate PDF files related to specified book titles.
- **Ensure Reliable PDF Retrieval:** Utilize web scraping techniques to extract PDF download links from search engine results.
- **Enhance Download Management:** Provide robust download management features including progress tracking and cancellation.

METHODOLY USED

- **Development Approach:** Adopted an Agile development methodology for iterative enhancements and quick adaptation to user feedback.
- **Programming Language:** Developed using Python for its versatility and extensive libraries, including Tkinter for GUI development.
- **Web Scraping:** Utilized the ‘requests’ library for HTTP requests and ‘BeautifulSoup’ for parsing HTML to extract PDF download links.
- **User Interface Design:** Implemented using Tkinter to create a responsive and intuitive GUI for seamless user interaction.

FLOWCHART


User Enters Book Name
Is Book Name Empty?
Yes: Show Error Message
No: Search for Book PDF
Search on Multiple Search Engines
Is PDF Found?
Yes: Prompt User to Select Save Location
No: Show Error Message
Is Save Location Selected?
No: Cancel Download
Yes: Download PDF with Progress Bar
Yes: Show Cancel Message
Is Download Cancelled?
Show Success Message
No: Save PDF to Selected Location

RESULTS AND OUTPUTS

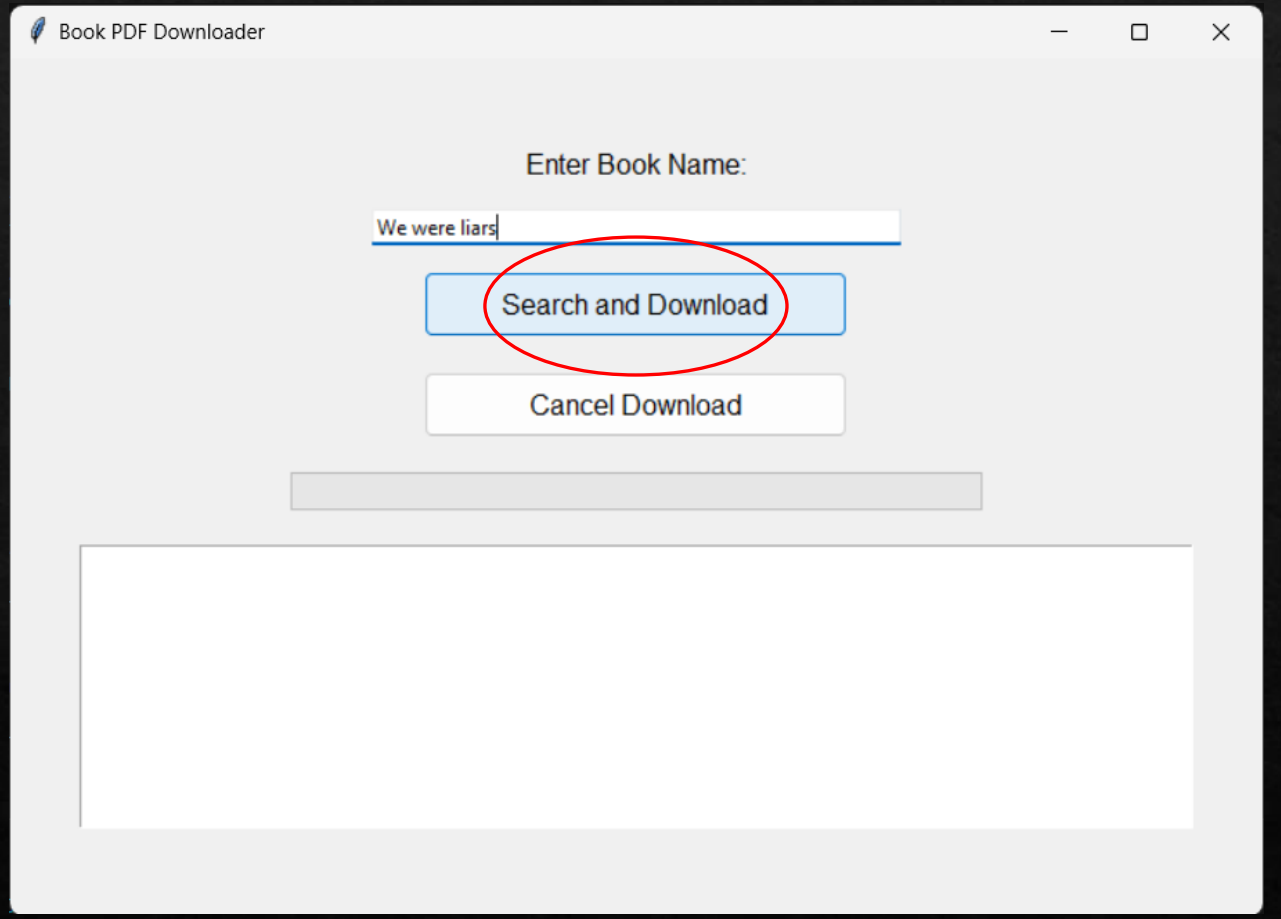
```python
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
import requests
from bs4 import BeautifulSoup
import os
import re
import threading

# Global variable to manage download cancellation
cancel_download_event = threading.Event()

def search_book():

    book_name = entry.get()
    if not book_name:
        messagebox.showwarning("Input Error", "Please enter a book name")
        return

    # Clear the cancel event before starting a new download
    cancel_download_event.clear()

    # Call the function to search and download the book PDF in a separate thread
    threading.Thread(target=download_book_pdf, args=(book_name,)).start()

def cancel_download():

    cancel_download_event.set()

def download_book_pdf(book_name):

    # Clear the log text widget
    log_text.delete(1.0, tk.END)

    log_text.insert(tk.END, f"Searching for '{book_name}'...\n")

    search_urls = [
        f"https://www.google.com/search?q={book_name.replace(' ', '+')}+filetype:pdf",
```

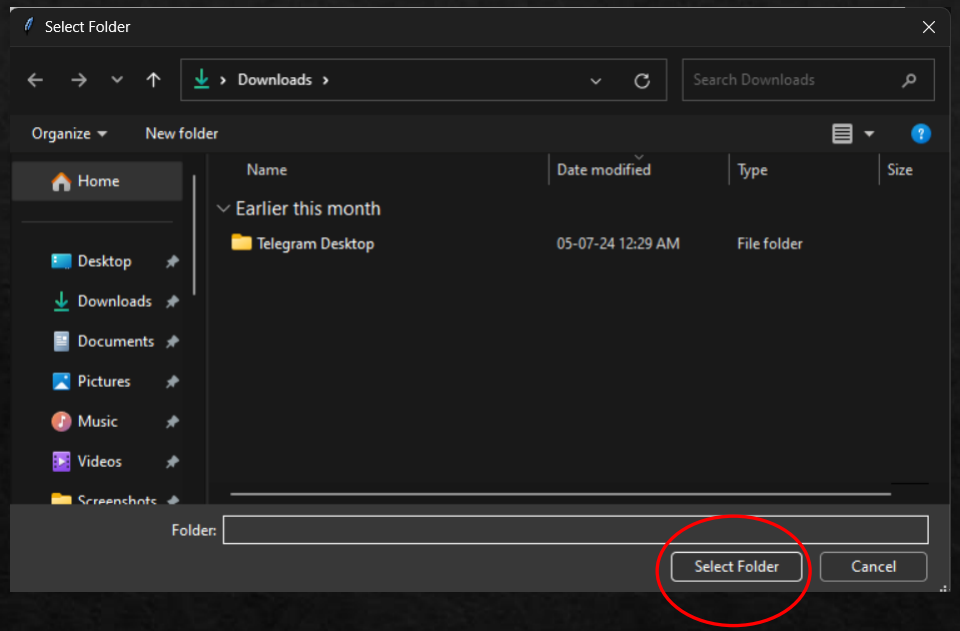
```
        f"https://www.bing.com/search?q={book_name.replace(' ', '+')}+filetype:pdf",
        f"https://www.duckduckgo.com/?q={book_name.replace(' ', '+')}+filetype:pdf"
    ]

    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
    }

    pdf_url = None
    for url in search_urls:
        log_text.insert(tk.END, f"Searching on {url}...\n")
        try:
            response = requests.get(url, headers=headers)
            soup = BeautifulSoup(response.text, "html.parser")

            for link in soup.find_all("a", href=True):
                href = link["href"]
                if ".pdf" in href:
                    pdf_url = href
                    break
            if pdf_url:
                break
        except Exception as e:
            log_text.insert(tk.END, f"Error while searching: {str(e)}\n")
            messagebox.showerror("Error", f"Error while searching: {str(e)}")
            return

    if not pdf_url:
        log_text.insert(tk.END, "No PDF found for the given book name\n")
        messagebox.showerror("Error", "No PDF found for the given book name")
        return

    log_text.insert(tk.END, f"PDF found at: {pdf_url}\n")

    # Prompt the user to select the download location
    save_path = filedialog.askdirectory()
    if not save_path:
```

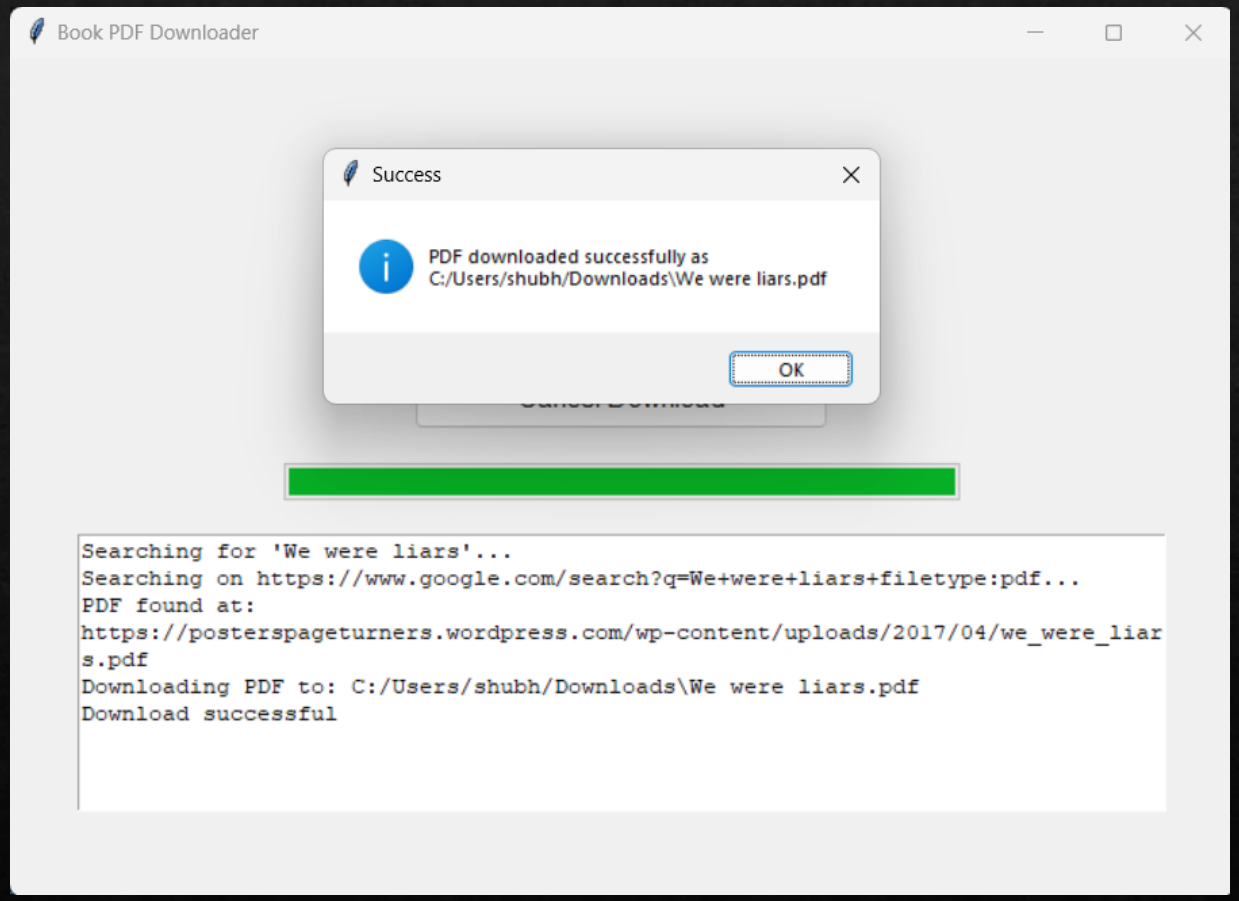
```python
    if not save_path:
        return

    pdf_name = re.sub(r'[\\/*?:"<>|]', "", book_name) + ".pdf"
    file_path = os.path.join(save_path, pdf_name)

    log_text.insert(tk.END, f"Downloading PDF to: {file_path}\n")

    # Download the PDF with progress bar
    try:
        response = requests.get(pdf_url, stream=True)
        total_size = int(response.headers.get('content-length', 0))

        progress_bar['value'] = 0
        progress_bar['maximum'] = total_size

        with open(file_path, 'wb') as pdf_file:
            for data in response.iter_content(chunk_size=4096):
                if cancel_download_event.is_set():
                    log_text.insert(tk.END, "Download cancelled\n")
                    messagebox.showinfo("Download Cancelled", "The download has been cancelled.")
                    return

                pdf_file.write(data)
                progress_bar['value'] += len(data)
                root.update_idletasks()

        log_text.insert(tk.END, "Download successful\n")
        messagebox.showinfo("Success", f"PDF downloaded successfully as {file_path}")
    except Exception as e:
        log_text.insert(tk.END, f"Failed to download PDF: {str(e)}\n")
        messagebox.showerror("Error", f"Failed to download PDF: {str(e)}")

# Create the main application window
root = tk.Tk()
root.title("Book PDF Downloader")

```

```
110
111 # Add padding to the main window
112 root.configure(padx=20, pady=20)
113
114 # Create a style
115 style = ttk.Style()
116 style.configure("TLabel", font=("Helvetica", 12))
117 style.configure("TEntry", font=("Helvetica", 12))
118 style.configure("TButton", font=("Helvetica", 12), padding=6)
119 style.configure("TProgressbar", thickness=20)
120
121 # Create and place the widgets inside a frame
122 frame = ttk.Frame(root, padding=(20, 20, 20, 20))
123 frame.pack(fill="both", expand=True)
124
125 label = ttk.Label(frame, text="Enter Book Name:")
126 label.pack(pady=10)
127
128 entry = ttk.Entry(frame, width=50)
129 entry.pack(pady=5)
130
131 search_button = ttk.Button(frame, text="Search and Download", command=search_book)
132 search_button.pack(pady=10)
133 search_button.configure(width=25)  # Adjust the button width
134
135 # Add a cancel button
136 cancel_button = ttk.Button(frame, text="Cancel Download", command=cancel_download)
137 cancel_button.pack(pady=10)
138 cancel_button.configure(width=25)  # Adjust the button width
139
140 # Add a progress bar
141 progress_bar = ttk.Progressbar(frame, orient='horizontal', length=400, mode='determinate')
142 progress_bar.pack(pady=10)
143
144 # Add a log text widget
145 log_text = tk.Text(frame, height=10, wrap=tk.WORD)
146 log_text.pack(pady=10, fill=tk.BOTH, expand=True)
147
148 # Run the application
149 root.mainloop()
150
```

OUTPUTS


Book PDF Downloader
Enter Book Name:
We were liars
Search and Download
Cancel Download


Select Folder
Downloads
Search Downloads
Organize
New folder
Home
Desktop
Downloads
Documents
Pictures
Music
Videos
Name
Date modified
Type
Size
Earlier this month
Telegram Desktop
05-07-24 12:29 AM
File folder
Folder:
Select Folder
Cancel


Book PDF Downloader
Success
PDF downloaded successfully as
C:/Users/shubh/Downloads\We were liars.pdf
OK
Searching for 'We were liars'...
Searching on https://www.google.com/search?q=We+were+liars+filetype:pdf...
PDF found at:
https://posterspageturners.wordpress.com/wp-content/uploads/2017/04/we_were_liars.pdf
Downloading PDF to: C:/Users/shubh/Downloads\We were liars.pdf
Download successful

FUTURE SCOPE

- Enhance the search functionality by refining search queries or using additional search engines or APIs.
- Provide feedback on the progress of the search operation itself (e.g., "Searching on Google/Bing/DuckDuckGo...").
- Add a mechanism to handle cases where multiple PDF links are found and allow the user to choose.
- Radio buttons or a dropdown menu to select between "PDF" and "Audiobook".
- Modify search URLs to include audiobook-specific queries.

CONCLUSION

Current Features

- Search and download PDFs of books based on user input.
- Progress tracking with a visual progress bar.
- Error handling for network issues and search failures.

Future Enhancements:

- Audiobook Integration: Extend functionality to download audiobooks alongside PDFs.
- Enhanced User Interface: Improve user experience with clearer feedback and options.
- Advanced Search: Implement advanced search algorithms or APIs for better results.

Benefits:

- Empowers users to access books in multiple formats effortlessly.
- Enhances learning and enjoyment through versatile content delivery.
- Vision: Continuously evolve to meet user needs and technological advancements in digital content access.

REFERENCES

- https://www.geeksforgeeks.org/
- https://docs.python.org/3/library/tkinter.html
- https://pypi.org/project/beautifulsoup4/
- https://docs.python.org/3/library/threading.html


Thank
you