**B.Bhargavi**

**07/06/2025**

**PDF Merger and Splitter - Code Documentation**

* **Overview**

This documentation describes the Python implementation of a PDF manipulation tool with GUI that provides two main functionalities:

1. Merging multiple PDF files into a single document

2. Splitting a single PDF file into individual pages

* **Table of Contents**

1. [Class Structure](#class-structure)

2. [Merge Functionality](#merge-functionality)

3. [Split Functionality](#split-functionality)

4. [GUI Components](#gui-components)

5. [Error Handling](#error-handling)

6. [Dependencies](#dependencies)

* **Class Structure**
* PDFToolApp

The main application class that implements all functionality.

* Initialization
* python

def \_\_init\_\_(self, root):

self.root = root

self.root.title("PDF Merger and Splitter")

self.root.geometry("600x500")

* GUI setup
* Key Methods

| Method | Description |

|--------|-------------|

| `setup\_merge\_tab()` | Creates the UI components for PDF merging |

| `setup\_split\_tab()` | Creates the UI components for PDF splitting |

| `add\_pdfs()` | Opens file dialog to select PDFs for merging |

| `remove\_selected()` | Removes selected files from merge list |

| `move\_item()` | Reorders files in merge list |

| `merge\_pdfs()` | Performs the PDF merging operation |

| `select\_split\_file()` | Selects PDF file to split |

| `select\_output\_folder()` | Chooses folder for split pages |

| `split\_pdf()` | Performs the PDF splitting operation |

**Merge Functionality**

* Workflow

1. User adds PDF files via "Add PDFs" button

2. Files appear in the listbox

3. User can reorder or remove files

4. User specifies output filename

5. On merge:

- Creates `PdfMerger` instance

- Appends all selected PDFs in order

- Writes to output file

**Key Code**

merger = PdfMerger()

for i in range(self.merge\_listbox.size()):

pdf\_path = self.merge\_listbox.get(i)

merger.append(pdf\_path)

with open(save\_path, 'wb') as f:

merger.write(f)

**Split Functionality**

* **Workflow**

1. User selects PDF file to split

2. Chooses output directory

3. (Optional) Sets filename prefix

4. On split:

- Creates `PdfReader` for input file

- Iterates through pages

- Creates new PDF for each page

- Saves with sequential numbering

**Key Code**

reader = PdfReader(f)

total\_pages = len(reader.pages)

for i in range(total\_pages):

writer = PdfWriter()

writer.add\_page(reader.pages[i])

output\_path = os.path.join(output\_folder, f"{prefix}{i+1}.pdf")

with open(output\_path, 'wb') as out\_f:

**GUI Components**

Merge Tab Components

| Component | Type | Purpose |

|-----------|------|---------|

| merge\_listbox | Listbox | Displays selected PDFs for merging |

| output\_name | Entry | Output filename input |

| Add PDFs | Button | Opens file selection dialog |

| Remove Selected | Button | Removes highlighted files |

| Move Up/Down | Buttons | Reorders files |

| Merge PDFs | Button | Executes merge operation |

**Split Tab Components**

| Component | Type | Purpose |

|-----------|------|---------|

| split\_file\_label | Label | Shows selected PDF path |

| output\_folder | Entry | Output directory path |

| file\_prefix | Entry | Prefix for split files |

| Select PDF | Button | Chooses PDF to split |

| Browse | Button | Chooses output folder |

| Split PDF | Button | Executes split operation |

**Error Handling**

The application includes comprehensive error handling:

1. Input Validation

- Checks for empty file selections

- Verifies output paths are specified

- Validates PDF file integrity

2. Exception Handling

- Catches file operations errors

- Handles PDF processing errors

- Provides user-friendly error messages

Example:

try:

# PDF operation code

except Exception as e:

messagebox.showerror("Error", f"Failed to merge PDFs:\n{str(e)}")

**Dependencies**

**Required Packages**

| Package | Purpose | Version |

|---------|---------|---------|

| PyPDF2 | PDF manipulation | 2.12.1+ |

| tkinter | GUI framework | Built-in |

**Installation**

Bash command

pip install PyPDF2

**Code Structure**

Pdfsplitter.py

├── \_\_init\_\_() - Initializes GUI

├── setup\_merge\_tab() - Merge UI setup

│ ├── add\_pdfs()

│ ├── remove\_selected()

│ ├── move\_item()

│ └── merge\_pdfs()

├── setup\_split\_tab() - Split UI setup

│ ├── select\_split\_file()

│ ├── select\_output\_folder()

│ └── split\_pdf()

└── Main execution block

**Best Practices**

1. Resource Management

- Uses context managers (`with` statements) for file operations

- Properly closes PDF merger objects

2.User Experience

- Clear status messages

- Intuitive file selection

- Visual feedback for operations

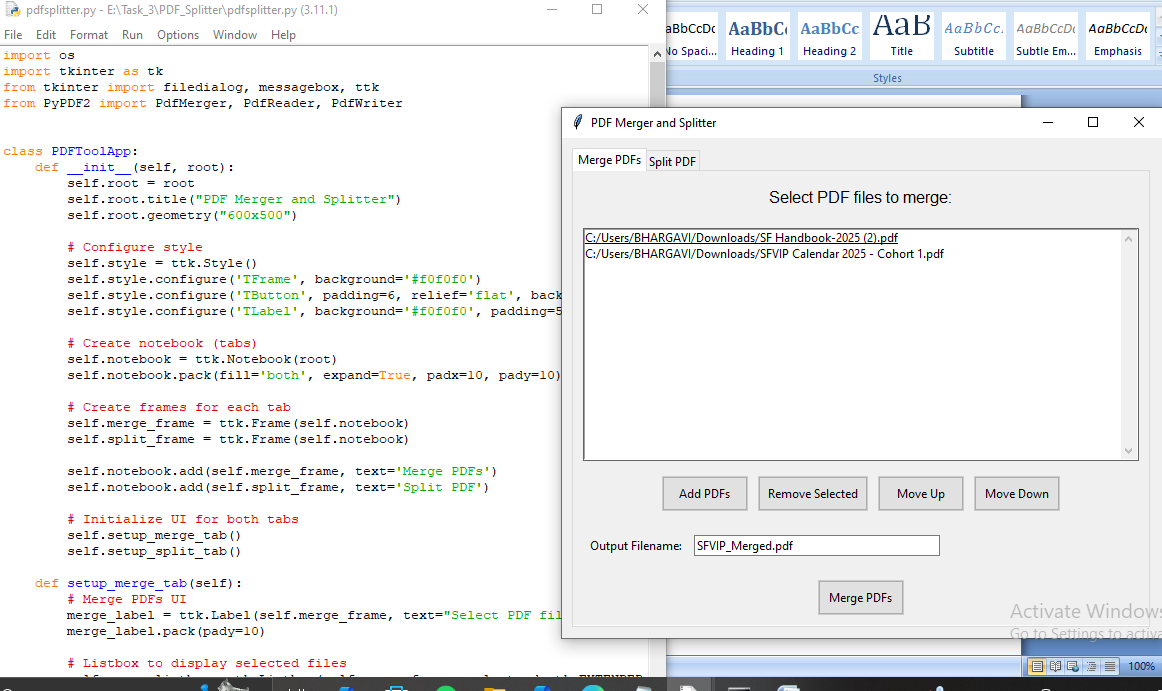
3. Modular Design

- Separate methods for distinct operations

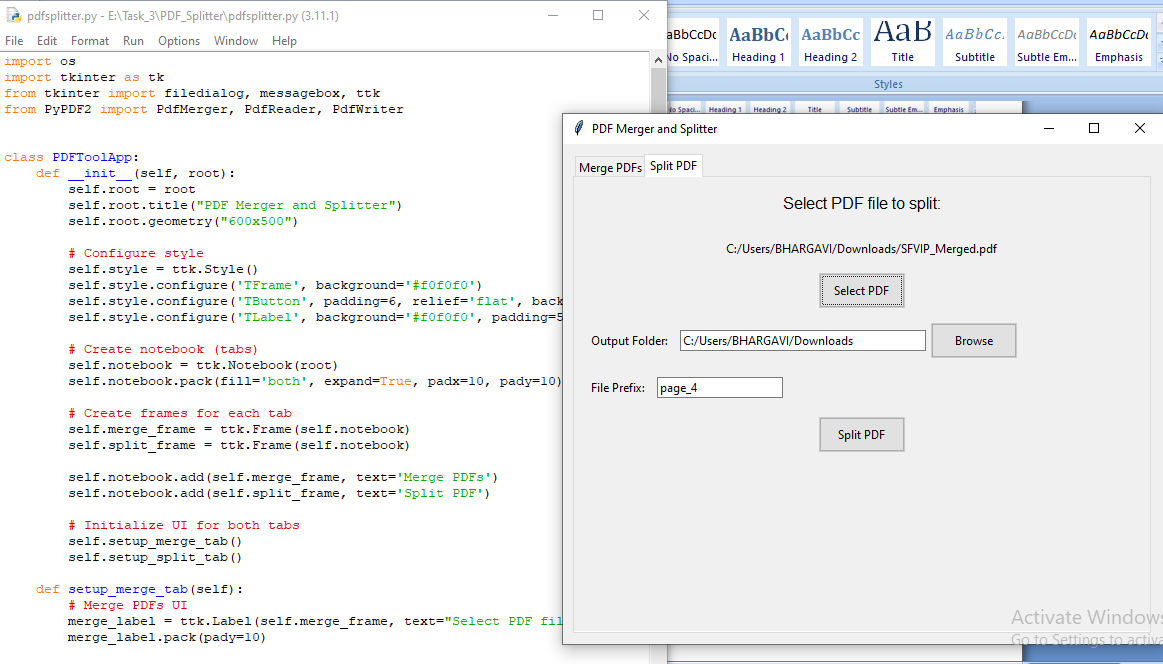
- Clear separation between UI and business logic

**Screenshots**

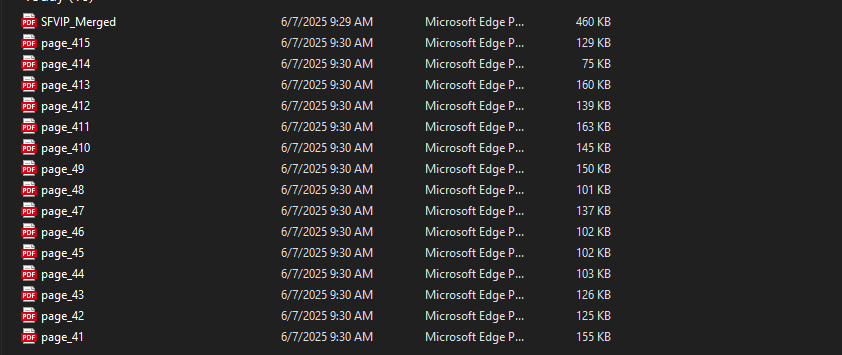
Mergepdf tab

****

Splitpdf tab



Merged and split operations performed o/p



**Extension Points**

1. Additional Features

- PDF rotation

- Password protection

- Page range selection for splitting

2. Enhancements

- Drag-and-drop file support

- Batch processing

- Progress indicators for large operations