1. Project Structure

The project will consist of the following components:

User Interface: A console-based interface for user interaction.

Functionality:

Merge multiple PDF files into one.

Split a single PDF into individual pages.

Error Handling: Handle invalid inputs (e.g., non-PDF files, missing files).

Reports: Generate logs of operations performed (optional).

Documentation: Provide clear instructions for setup and usage.

2. Technologies

Python: Core programming language.

PyPDF2: Library for working with PDF files.

Logging: For generating logs of operations.

Tkinter (Optional): For a GUI-based interface.

SQLite (Optional): For storing operation history.

3. Implementation Steps

Step 1: Install Required Libraries

Install the necessary Python libraries:

```
pip install PyPDF2
```

Step 2: Core Functionality

Implement the core functionality for merging and splitting PDFs.

```
import os
from PyPDF2 import PdfMerger, PdfReader, PdfWriter


def merge_pdfs(input_files, output_file):
    """
    Merge multiple PDF files into one.
    :param input_files: List of input PDF file paths.
    :param output_file: Output PDF file path.
    """
    merger = PdfMerger()
    for file in input_files:
        if not file.endswith('.pdf'):
            raise ValueError(f"Invalid file type: {file}. Only PDF files are supported.")
        merger.append(file)
    merger.write(output_file)
    merger.close()
    print(f"Merged PDF saved as {output_file}")


def split_pdf(input_file, output_folder):
    """
    Split a single PDF into individual pages.
    :param input_file: Input PDF file path.
    :param output_folder: Folder to save individual pages.
    """
    if not input_file.endswith('.pdf'):
        raise ValueError(f"Invalid file type: {input_file}. Only PDF files are supported.")

    reader = PdfReader(input_file)
    os.makedirs(output_folder, exist_ok=True)
    for i, page in enumerate(reader.pages):
        writer = PdfWriter()
        writer.add_page(page)
        output_file = os.path.join(output_folder, f"page_{i + 1}.pdf")
        with open(output_file, "wb") as out:
            writer.write(out)
        print(f"Page {i + 1} saved as {output_file}")
```

Step 3: Console-Based User Interface

Create a simple console-based interface for user interaction.

```
import logging


# Configure logging
logging.basicConfig(filename="pdf_operations.log", level=logging.INFO, format="%(asctime)s - %(message)s")


def main():
    print("PDF Merger and Splitter")
    print("1. Merge PDFs")
    print("2. Split PDF")
    choice = input("Enter your choice (1 or 2): ")

    if choice == "1":
        input_files = input("Enter the paths of the PDF files to merge (comma-separated): ").split(",")
        output_file = input("Enter the output file name (e.g., merged.pdf): ")
        try:
            merge_pdfs(input_files, output_file)
            logging.info(f"Merged PDFs: {input_files} into {output_file}")
        except Exception as e:
            print(f"Error: {e}")
            logging.error(f"Error merging PDFs: {e}")
    elif choice == "2":
        input_file = input("Enter the path of the PDF file to split: ")
        output_folder = input("Enter the output folder name: ")
        try:
            split_pdf(input_file, output_folder)
            logging.info(f"Split PDF: {input_file} into {output_folder}")
        except Exception as e:
            print(f"Error: {e}")
            logging.error(f"Error splitting PDF: {e}")

    else:
        print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Step 4: Optional Features

- **GUI Using Tkinter**: Create a graphical interface for better user experience.
- **SQLite for History**: Store operation history in a SQLite database.
- **Pandas for Reports**: Generate reports in table format.

4. Testing

Test the application for various edge cases:

- Non-PDF files as input.
- Missing files.
- Invalid user inputs.

5. Deliverables

1. **Source Code**: Python script(s) for the application.
2. **Executable**: Use pyinstaller to create an executable (optional).

```
pip install pyinstaller

pyinstaller --onefile pdf_tool.py
```

1. **Documentation**:

- **Setup Instructions**: How to install and run the application.
- **Usage Guide**: How to use the application for merging and splitting PDFs.
- **Logs**: Explanation of the log file (pdf_operations.log).

6. Example Documentation

Setup Instructions

1. Install Python 3.x from python.org.
2. Install the required libraries:

```
pip install PyPDF2
```

3.Download the pdf_tool.py script.

1. Run the script:

```
python pdf_tool.py
```

2. Follow the on-screen instructions to merge or split PDFs.