INTRODUCTION

All of you must be familiar with what PDFs are. In-fact, they are one of the most important and widely used digital media. PDF stands for **Portable Document Format**. It uses **.pdf** extension. It is used to present and exchange documents reliably, independent of software, hardware, or operating system.
Invented by **Adobe**, PDF is now an open standard maintained by the International Organization for Standardization (ISO). PDFs can contain links and buttons, form fields, audio, video, and business logic.

The project PDF Extractor will help people to extract PDF file into simple text even the Snapshot of any hand written page can be extracted which may help in getting it converted into text document for further use and also this application can be used for rotating ,merging, splitting the PDF as well .

This is designed in one language Python as front-end and the back-end.

REALISATION OF THE PROBLEM

Nowadays, existing system lack proper GUI (Graphical User Interface) and that system can only be executed by trained programmer or the person who have knowledge of programming , no other user can able to use the system

OBJECTIVES

The new PDF EXTRACTOR system will eradicate the issues that they are facing now and this software is created in such a manner that it can be used by anyone , even a person who have just a little knowledge of working of computer.
It also gives us a path to wherever we want to save to file which lacks in the previous system as well as the option to select the desired file.

PROBLEM FORMULATION

Problem Definition-

PDF EXTRACTOR system will eradicate the issues that they are facing now and this software is created in such a manner that it can be used by anyone , even a person who have just a little knowledge of working of computer.

It also gives us a path to wherever we want to save to file which lacks in the previous system as well as the option to select the desired file.

The project PDF Extractor will help people to extract PDF file into simple text even the Snapshot of any hand written page can be extracted which may help in getting it converted into text document for further use and also this application can be used for rotating ,merging, splitting the PDF as well.

Proposed System-

The proposed system is having many advantages over the existing system. It requires less overhead and is very efficient. The new proposed PDF EXTRACTOR system will eradicate the issues that they are facing now and this software is created in such a manner that it can be used by anyone , even a person who have just a little knowledge of working of computer.

It also gives us a path to wherever we want to save to file which lacks in the previous system as well as the option to select the desired file

REQUIREMENT SPECIFICATION

Performance Requirement-

Performance of the application will highly depend upon the performance of hardware and software components of the computer. Coming to timing relationships, the load time for user interface screens should not take longer than a seconds.

Safety Requirement-

- Application will not cause any harm to human users.
- Application will be restored in case of any emergency.

Security Requirement-

Application is designed in such a way that it cannot use file of some other application.
Different modules work in co-ordination with other modules, they can’t work independently.

FEASIBILITY STUDY

The prime focus of the feasibility is evaluating the practicality of the proposed system keeping in mind a number of factors. The following factors are taken into account before deciding in favor of the new system. It is of three types:

- Economic Feasibility
- Technical Feasibility
- Operational feasibility

Economic Feasibility-

A project is considered economically feasible when the benefits are greater than the cost of undertaking the project. This project is going to be economically feasible as it is going to reduce the manual work done on paper thereby reducing the cost incurred in paper.

Technical feasibility-

The technical feasibility study describes the details of how you will deliver a product or service i.e. what technology and expertise are required.

Keeping in view the above fact, nowadays all organizations are automating the repetitive and monotonous works done by humans. The key process areas of the current system are nicely amenable to automation and hence the technical feasibility is proved beyond doubt.

Operational Feasibility-

The present system has automated most of the manual tasks. Therefore, the proposed system will increase the operational efficiency of the transport system. This system is a detection system, which needs small amount of resources. Department of transportation or security can meet the conditions both in hardware and software aspects; therefore, this system is feasible in operation.

SYSTEM ANALYSIS AND DESIGN

Description Of Module-

- **USER**

  When user login to the system then following activities can be performed-

  1. Can upload PDF.
  2. Get the image PDF converted to doc or in text format.
  3. Get the PDF file rotated , merged or spitted.

SYSTEM IMPLEMENTATION

Hardware Specifications-

• Processor:	Intel Core I3 or higher power
• RAM:	2 GB or more
• Hard disk:	Around 1 GB or more
• Keyboard:	Normal or Multimedia
• Mouse:	Compatible Mouse

Software Specifications-

- Operating System: This application can run on Linux, Windows and Mac or any other OS in which “OpenCV” (a python library) is preinstalled.

RESULT AND DISCUSSION

Future Scope Of the Project-

“PDF EXTRACTOR” is an application which has a wide range of scope. This project was developed to fulfill the requirements, however there are lots of scope to improve the performance of the system in the area of user interface, data set, etc.

Therefore, there are many features for future enhancement of this project. One of the future enhancements of the project are as follows-

- It may be integrated with other software’s and applications to provide additional functionalities and features. For instance, the extracted text may be sent to another application that has a database, and it may be matched with that database for vehicle identification or tracking.

Importance of Work -

The modern technology has proved that machine is much efficient and reliable than man. The computer has made great revolution in the world and we can say that it is the best creation of the man in the 20th century. The computer gets the popularity due to its fast processing of data, reliability, and storage capacity.

<u>Advantages And Special Features Of The System-</u>

The project PDF Extractor will help people to extract PDF file into simple text even the Snapshot of any hand written page can be extracted which may help in getting it converted into text document for further use and also this application can be used for rotating ,merging, splitting the PDF as well.

<u>Limitations</u> -

No software in the world can be called perfect as it has some or the other flaw or lacking, that is new version keeps on pouring. Similarly, this system also possess some limitations such as this system will not work for images in dark.

CONCLUSION

The project “PDF EXTRACTOR” is completed, satisfying the required design specifications. The system provides a user-friendly interface. The software is developed with modular approach. All modules in the system have been tested with valid data and invalid data and everything work successfully. Thus, the system has fulfilled all the objectives identified and is able to replace the existing system. The constraints are met and overcome successfully. The system is designed as like it was decided in the design phase. This software has a user-friendly screen that enables the user to use without any inconvenience.

The software is a stand-alone, self-contained program and is not a part of a product family.

REFERENCE/BIBLIOGRAPHY

1. Pulli, Kari; Baksheev, Anatoly; Kornyakov, Kirill; Eruhimov, Victor (1 April 2012). "Realtime Computer Vision with OpenCV". Queue: 40:40–40:56. doi:10.1145/2181796.2206309(inactive 2019-08-19).
2. Intel acquires Itseez: https://opencv.org/intel-acquires-itseez.html
3. https://github.com/opencv/opencv/wiki/Deep-Learning-in-OpenCV
4. Adrian Kaehler; Gary Bradski (14 December 2016). Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media. pp. 26ff. ISBN 978-1-4919-3800-3.
5. Bradski, Gary; Kaehler, Adrian (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc. p. 6.
6. OpenCV change logs: http://code.opencv.org/projects/opencv/wiki/ChangeLog
7. OpenCV Developer Site: http://code.opencv.org
8. OpenCV User Site: http://opencv.org/
9. "Intel Acquires Computer Vision for IOT, Automotive | Intel Newsroom". Intel Newsroom. Retrieved 2018-11-26.
10. "Intel acquires Russian computer

APPENDIX

Data flow diagram-

A Data Flow Diagram (DFD) is a diagram that describes the flow of data and the processes that change data throughout a system. It's a structured analysis and design tool that can be used for flowcharting in place of or in association with information oriented and process oriented system flowcharts. When analysts prepare the Data Flow Diagram, they specify the user needs at a level of detail that virtually determines the information flow into and out of the system and the required data resources. This network is constructed by using a set of symbols that do not imply physical implementations. The Data Flow Diagram reviews the current physical system, prepares input and output specification, specifies the implementation plan etc.

Four basic symbols are used to construct data flow diagrams. They are symbols that represent data source, data flows, and data transformations and data storage. The points at which data are transformed are represented by enclosed figures, usually circles, which are called nodes.

The Data flow Diagram shows the flow of data. It is generally made of symbols given below:-

A square shows the Entity.

A Circle shows the Process

An open Ended Rectangle shows the data store.

An arrow shows the data flow.

The DFD can be up to several levels. The 0 level DFD states the flow of data in the system as seen from the outward in each module.

The first level DFD show more detail, about the single process of 0 levels DFD

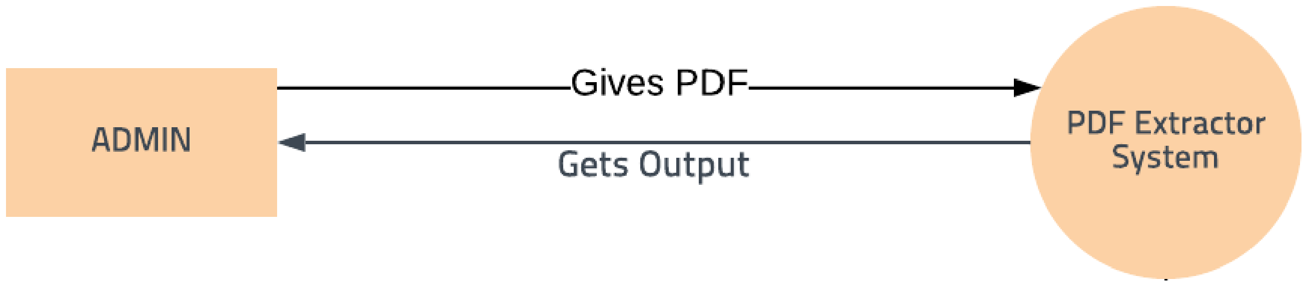


FIG: ZERO LEVEL DFD OF PDF EXTRACTOR

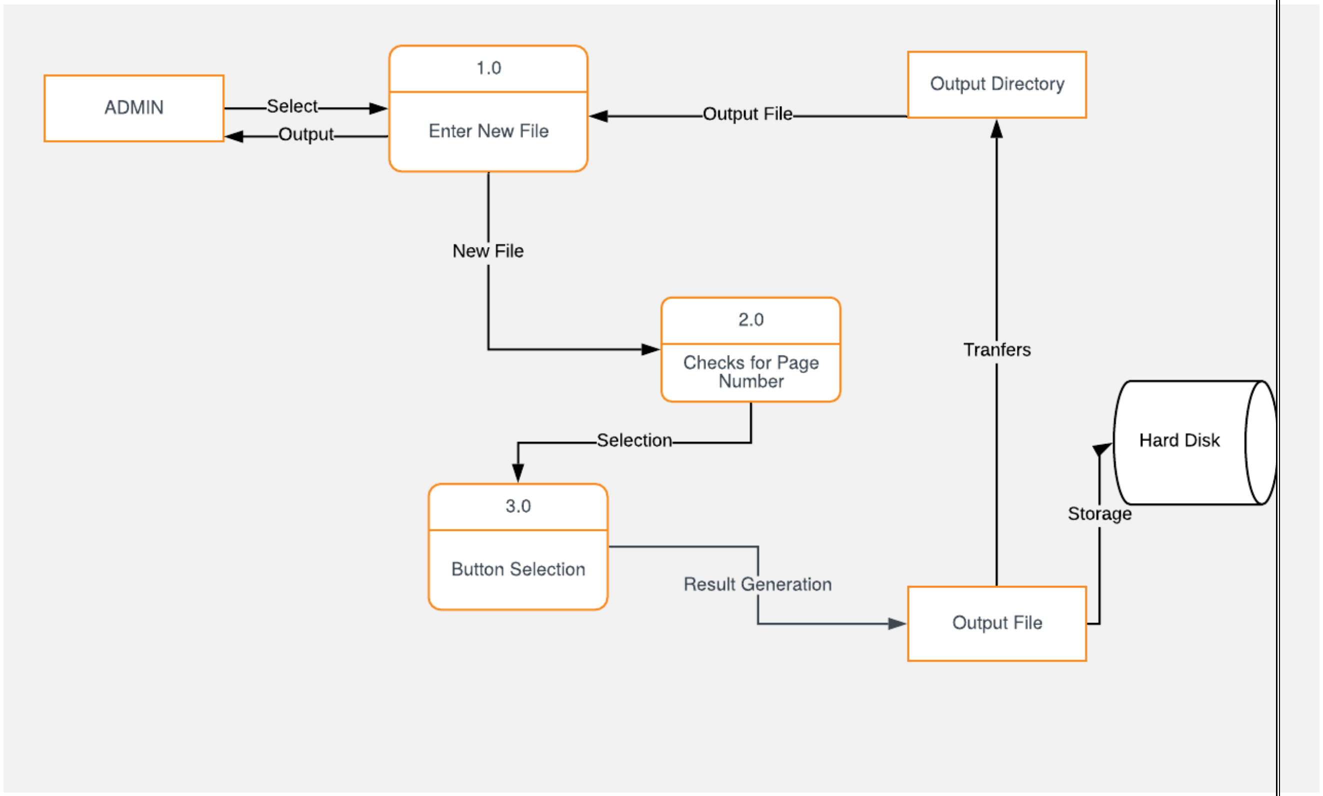


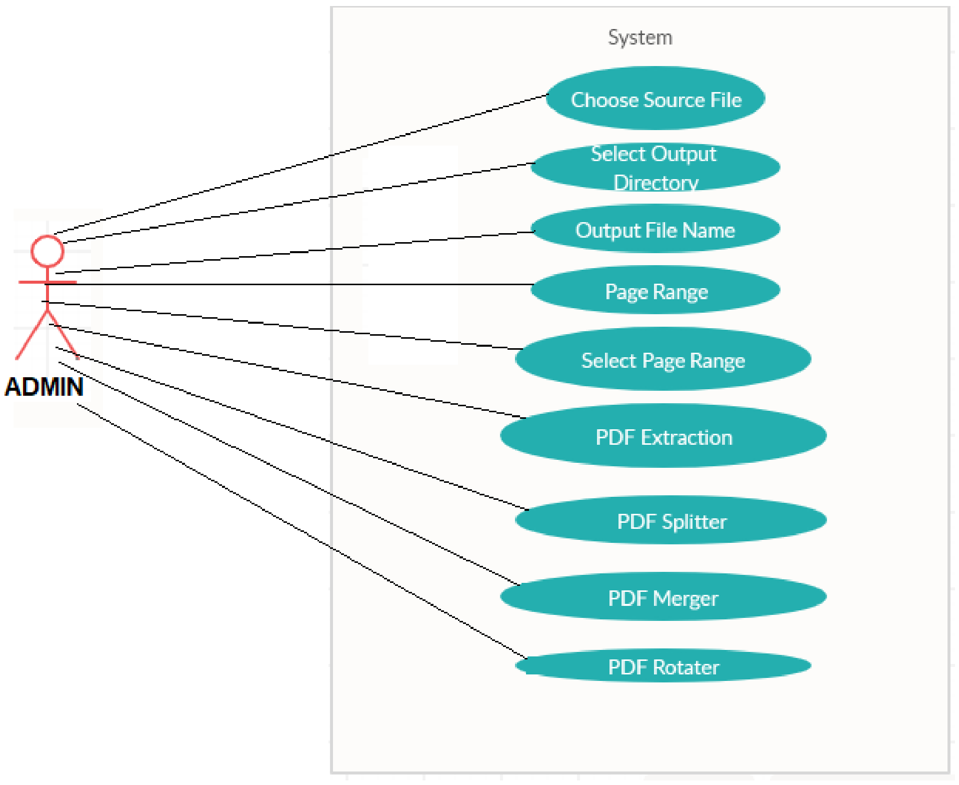
FIG: ONE LEVEL DFD OF PDF EXTRACTOR

USE CASE:

As the most known diagram type of the behavioral UML diagrams, Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.

It’s a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system

USE CASE:



SAMPLE CODING

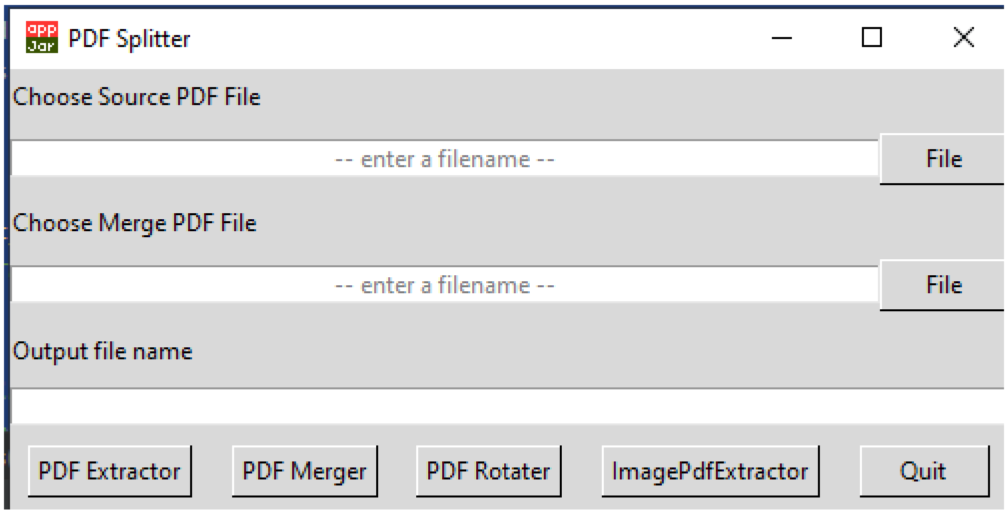
```
from appJar import gui
import PyPDF2
from PyPDF2 import PdfFileWriter, PdfFileReader
from pathlib import Path
from PyPDF2 import PdfFileMerger

import pytesseract
from PIL import Image
from wand.image import Image as wi
import os
# Define all the functions needed to process the files


def split_pages(input_file, page_range, out_file):
    """ Take a pdf file and copy a range of pages into a new pdf file

    Args:
        input_file: The source PDF file
        page_range: A string containing a range of pages to copy: 1-3,4
        out_file: File name for the destination PDF
    """
    output = PdfFileWriter()
    input_pdf = PdfFileReader(open(input_file, "rb"))
    output_file = open(out_file, "wb")

    # https://stackoverflow.com/questions/5704931/parse-string-of-integer-sets-with-intervals-to-list
    page_ranges = (x.split("-") for x in page_range.split(","))
    range_list = [i for r in page_ranges for i in range(int(r[0]), int(r[-1]) + 1)]

    for p in range_list:
        # Need to subtract 1 because pages are 0 indexed
        try:
            output.addPage(input_pdf.getPage(p - 1))
        except IndexError:
            # Alert the user and stop adding pages
            app.infoBox("Info", "Range exceeded number of pages in input.\nFile will still be saved.")
            break
    output.write(output_file)

    if(app.questionBox("File Save", "Output PDF saved. Do you want to quit?")):
        app.stop()
```

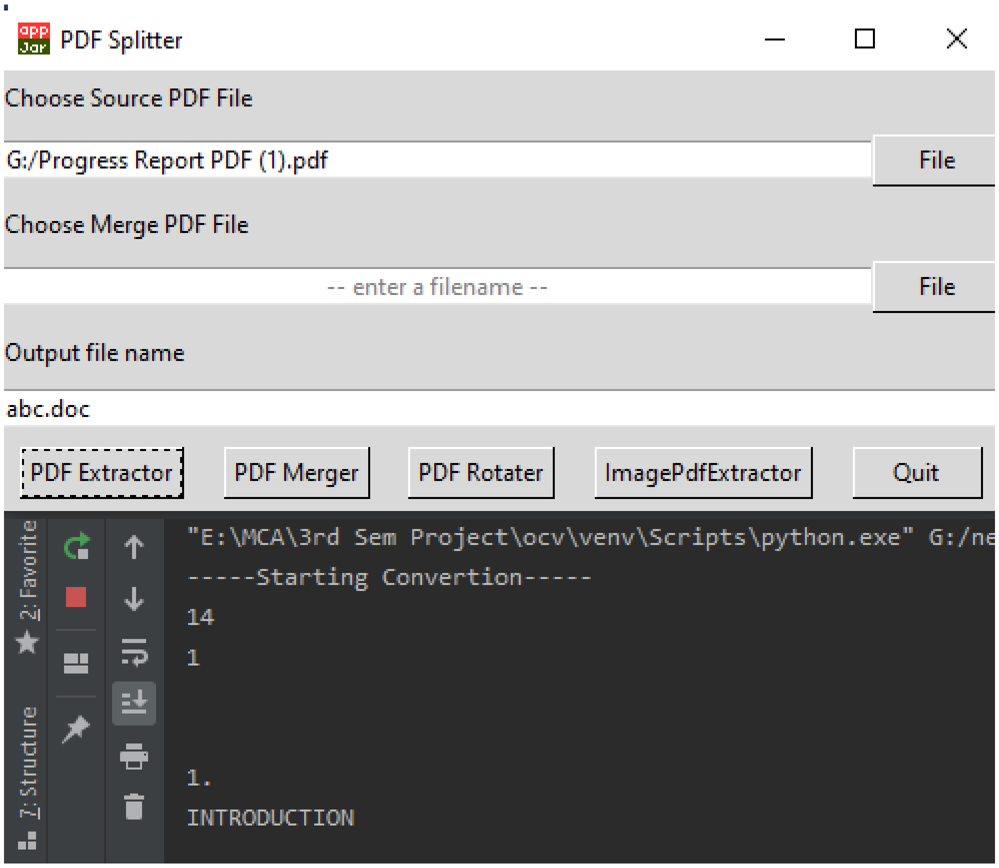
```
def validate_inputs(input_file, output_dir, range, file_name):
    """ Verify that the input values provided by the user are valid

    Args:
        input_file: The source PDF file
        output_dir: Directory to store the completed file
        range: File A string containing a range of pages to copy: 1-3,4
        file_name: Output name for the resulting PDF

    Returns:
        True if error and False otherwise
        List of error messages
    """
    errors = False
    error_msgs = []

    # Make sure a PDF is selected
    if Path(input_file).suffix.upper() != ".PDF":
        errors = True
        error_msgs.append("Please select a PDF input file")

    # Make sure a range is selected
    if len(range) < 1:
        errors = True
        error_msgs.append("Please enter a valid page range")

    # Check for a valid directory
    if not(Path(output_dir)).exists():
        errors = True
        error_msgs.append("Please Select a valid output directory")

    # Check for a file name
    if len(file_name) < 1:
        errors = True
        error_msgs.append("Please enter a file name")

    return(errors, error_msgs)


def press(button):
    """ Process a button press

    Args:
        button: The name of the button. Either Process of Quit
    """
    if button == "Process":
        src_file = app.getEntry("Input_File")
        dest_dir = app.getEntry("Output_Directory")
        page_range = app.getEntry("Page_Ranges")
        out_file = app.getEntry("Output_name")
        errors, error_msg = validate_inputs(src_file, dest_dir, page_range, out_file)
        if errors:
            app.errorBox("Error", "\n".join(error_msg), parent=None)
        else:
            split_pages(src_file, page_range, Path(dest_dir, out_file))
    elif button == "PDF Extractor":
        src_file = app.getEntry("Input_File")
        out_file = app.getEntry("Output_name")

         # creating a pdf file object
        pdfFileObj = open(src_file, 'rb')

            # creating a pdf reader object
        pdfReader = PyPDF2.PdfFileReader(pdfFileObj)

            # printing number of pages in pdf file
        print(pdfReader.numPages)

            # creating a page object
        pageObj = pdfReader.getPage(0)
        f = open(out_file, "w+")
        f.write(pageObj.extractText())
            # extracting text from page
        print(pageObj.extractText())


            # closing the pdf file object
        pdfFileObj.close()
        f.close()
        if (app.questionBox("File Save", "Output PDF saved. Do you want to quit?")):
            app.stop()
    elif button == "PDF Merger":

        src_file = app.getEntry("Input_File")
        merge_file= app.getEntry("Merge_File")
        out_file = app.getEntry("Output_name")

        pdfs = [src_file, merge_file]

        merger = PdfFileMerger()

        for pdf in pdfs:
            merger.append(pdf)
        merger.write(out_file)
        merger.close()
        if (app.questionBox("File Save", "Output PDF saved. Do you want to quit?")):
            app.stop()
    elif button == "ImagePdfExtractor":
        src_file = app.getEntry("Input_File")
        out_file = app.getEntry("Output_name")
        print('-----Starting Convertion-----')

        pdf = wi(filename=src_file, resolution=300)
        pdfimage = pdf.convert("jpeg")
        i = 1
        for img in pdfimage.sequence:
            page = wi(image=img)
            page.save(filename=str(i) + ".jpg")
            i += 1

            pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files (x86)\Tesseract-
OCR\tesseract.exe"

        for x in range(1, 31):


            img = Image.open("C:\\OC\\{0}.jpg".format(x))

            text = pytesseract.image_to_string(img)
            print(text)
            f = open(out_file,'a+')
            f.write(text)
        print('-----Done-----')
        if (app.questionBox("File Save", "Output PDF saved. Do you want to quit?")):
            app.stop()
    elif button == "PDF Splitter":
        src_file = app.getEntry("Input_File")
        out_file = app.getEntry("Output_name")

        def PDFsplit(pdf, splits):
            # creating input pdf file object
            pdfFileObj = open(pdf, 'rb')

            # creating pdf reader object
            pdfReader = PyPDF2.PdfFileReader(pdfFileObj)

            # starting index of first slice
            start = 0

            # starting index of last slice
            end = splits[0]

            for i in range(len(splits) + 1):
                # creating pdf writer object for (i+1)th split
                pdfWriter = PyPDF2.PdfFileWriter()

                # output pdf file name
                outputpdf = pdf.split('.pdf')[0] + str(i) + '.pdf'

                # adding pages to pdf writer object
                for page in range(start, end):
                    pdfWriter.addPage(pdfReader.getPage(page))

                    # writing split pdf pages to pdf file
                with open(outputpdf, "wb") as f:
                    pdfWriter.write(f)

                    # interchanging page split start position for next split
                start = end
                try:
                    # setting split end position for next split
                    end = splits[i + 1]
                except IndexError:
                    # setting split end position for last split
                    end = pdfReader.numPages

                    # closing the input pdf file object
            pdfFileObj.close()
            if (app.questionBox("File Save", "Output PDF saved. Do you want to quit?")):
                app.stop()

        def main():
            # pdf file to split
            pdf = src_file
            # split page positions
            splits = [2, 4]

            # calling PDFsplit function to split pdf
            PDFsplit(pdf, splits)

        if __name__ == "__main__":
            # calling the main function
            main()
    elif button == "PDF Rotater":
        src_file = app.getEntry("Input_File")
```

```
def PDFrotate(origFileName, newFileName, rotation):

    # creating a pdf File object of original pdf
    pdfFileObj = open(origFileName, 'rb')

    # creating a pdf Reader object
    pdfReader = PyPDF2.PdfFileReader(pdfFileObj)

    # creating a pdf writer object for new pdf
    pdfWriter = PyPDF2.PdfFileWriter()

    # rotating each page
    for page in range(pdfReader.numPages):
        # creating rotated page object
        pageObj = pdfReader.getPage(page)
        pageObj.rotateClockwise(rotation)

        # adding rotated page object to pdf writer
        pdfWriter.addPage(pageObj)

        # new pdf file object
    newFile = open(newFileName, 'wb')

    # writing rotated pages to new file
    pdfWriter.write(newFile)

    # closing the original pdf file object
    pdfFileObj.close()

    # closing the new pdf file object
    newFile.close()

def main():

    # original pdf file name
    origFileName = src_file
    out_file = app.getEntry("Output_name")
    # new pdf file name
    newFileName = out_file

    # rotation angle
    rotation = 270

    # calling the PDFrotate function
    PDFrotate(origFileName, newFileName, rotation)

if __name__ == "__main__":
```

```
            # calling the main function
            main()
    else:
        app.stop()

# Create the GUI Window
app = gui("PDF Splitter", useTtk=True)
app.setTtkTheme("default")
app.setSize(500, 200)

# Add the interactive components
app.addLabel("Choose Source PDF File")
app.addFileEntry("Input_File")

app.addLabel("Choose Merge PDF File")
app.addFileEntry("Merge_File")

app.addLabel("Select Output Directory")
app.addDirectoryEntry("Output_Directory")

app.addLabel("Output file name")
app.addEntry("Output_name")

app.addLabel("Page Ranges: 1,3,4-10")
app.addEntry("Page_Ranges")

# link the buttons to the function called press
app.addButtons(["PDF Extractor","PDF Merger","PDF Rotater","ImagePdfExtractor", "Quit"], press)



# start the GUI
app.go()
```

OUTPUT SCREEN:

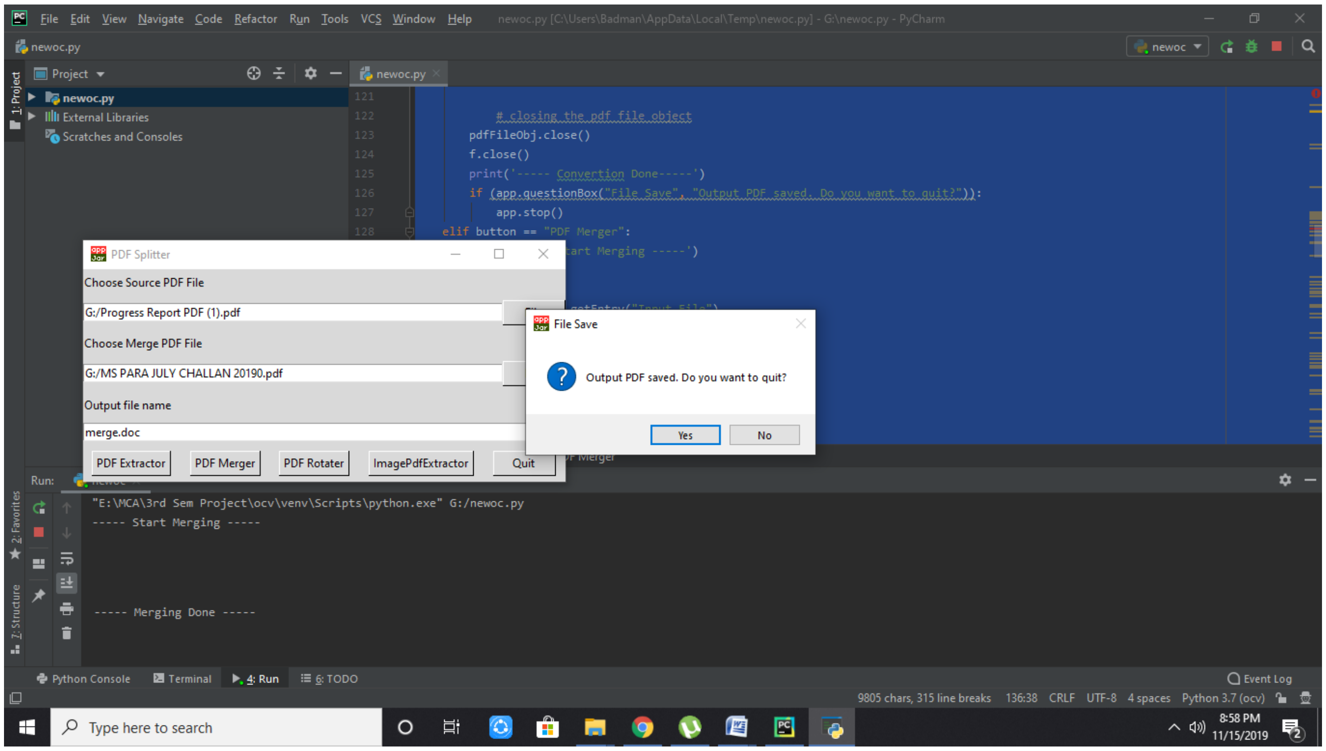




Choose Source PDF File: G:/Progress Report PDF (1).pdf

Output file name: abc.doc

```
"E:\MCA\3rd Sem Project\ocv\venv\Scripts\python.exe" G:/ne
-----Starting Convertion-----
14
1


1.
INTRODUCTION
```



newoc.py [C:\Users\Badman\AppData\Local\Temp\newoc.py] - G:\newoc.py - PyCharm

```
        # closing the pdf file object
    pdfFileObj.close()
    f.close()
    print('----- Convertion Done-----')
    if (app.questionBox("File Save", "Output PDF saved. Do you want to quit?")):
        app.stop()
elif button == "PDF Merger":
```

PDF Splitter

Choose Source PDF File: G:/Progress Report PDF (1).pdf

Choose Merge PDF File: G:/MS PARA JULY CHALLAN 20190.pdf

Output file name: merge.doc

File Save

Output PDF saved. Do you want to quit?

```
"E:\MCA\3rd Sem Project\ocv\venv\Scripts\python.exe" G:/newoc.py
----- Start Merging -----



----- Merging Done -----
```