SHAHEED BHAGAT SINGH STATE TECHNICAL CAMPUS

PROJECT REPORT

BACHELOR OF TECHNOLOGY

(Session-2018-2022)

SUBMITTED TO:

MRS. SONIKA JINDAL

SUBMITTED BY:

Abhijeet Ganguly (1809134)

Ajay Kumar(1809139)

Amit Kumar Mandal(1809143)

B.Tech. (cse) 6th sem

PDF CONVERTER

Authors:

Abhijeet Ganguly

Amit Kumar Mandal

Ajay Kumar

CONTENT

PREFACE

Due to Covid-19 the need for digital tool is at it’s high. One of the requirement is to provide a solution for online document submission. We thought a solution by providing a tool to overcome respective needs.

We have designed a pdf converter with various features like- IMG to PDF, WORD to PDF, PPT to PDF, PDF Merger, Text OCR etc. to overcome the needs of today’s digital environment. Also provides some other flexible tools too.

IMG to PDF – converts IMG files to a pdf

WORD to PDF – converts WORD files to pdf

PPT to PDF -converts PPT files to pdf

PDF Merger – Merges multiple pdf file in one

Text OCR – extracts text from IMG

ACKNOWLEDGEMENT

We are highly grateful to, HoD of Computer Science Engineering and Director of Shaheed Bhagat Singh State Technical Campus, Ferozepur for providing us the opportunity to carry out our Project activities. we express our sincere gratitude to those who have generously helped us in providing the valuable knowledge.
we would like to take this opportunity to convey our deep sense of gratitude to our External guide, for the direction, support and steady motivation.
At the end we would love to thank all of staff members for providing an excellent and healthy environment during time of project.
Finally, we express our indebtedness to all who have directly or indirectly contributed to the Successful completion of our project.

AMIT KUMAR MANDAL

ABHIJEET GANGULY

AJAY KUMAR

Description:

Pdf Converter

Project :

A pdf converter which processes images and word files to give output as pdf file. Also have features like pdf files merging, reodering pages etc.

Justification:

As due to COVID-19 most of the institutions are taking exams online (open book). But we observed that most of the students are facing problems with this mode of exam conducting due to insufficient knowledge about pdf conversion , or the complex interface of already available pdf converters , the limited size of pages in pdf etc.

Also , Teachers(mostly school teachers) are tensed too by this problem . As it is completely new environment for them too.

So , we thought to provide a solution to this problem. As it's a very common and widely observed problem.

Few Features:

OCR reader: read words from images .

Image to PDF: convert multiple image sources in a single compiled PDF file for easy management and avoid loses.

Cropping(to reduce noise): It is a side feature of image to Pdf which gives the ability to remove unnecessary parts of image.

Merging(different pdf files): It is a feature of pdf section which gives the ability to merge different pdf file together in required order.

Gaps identified in already present projects:

As we there are already products present in market similar to this problem. But they have their own limitation like:

- Most are the paid applications,
- a few of the free one's are not good designed,
- a few have page limitation ,
- Most of the application are online syn enabled in the app provider's cloud facilitate which is really a matter of concern for privacy & security.
- Complex design
- Time consuming

Objective :

To design a desktop-based pdf converter application which will be free of cost with most of the features, which are required for our problem satisfaction. Application with simple design.

Resources Needed:

Language: Python

Libraries:

Sys

Progress

Pillow/ PIL

Ttk

Cv2 os numpy img2pdf win32com pytesseract PyPdf2

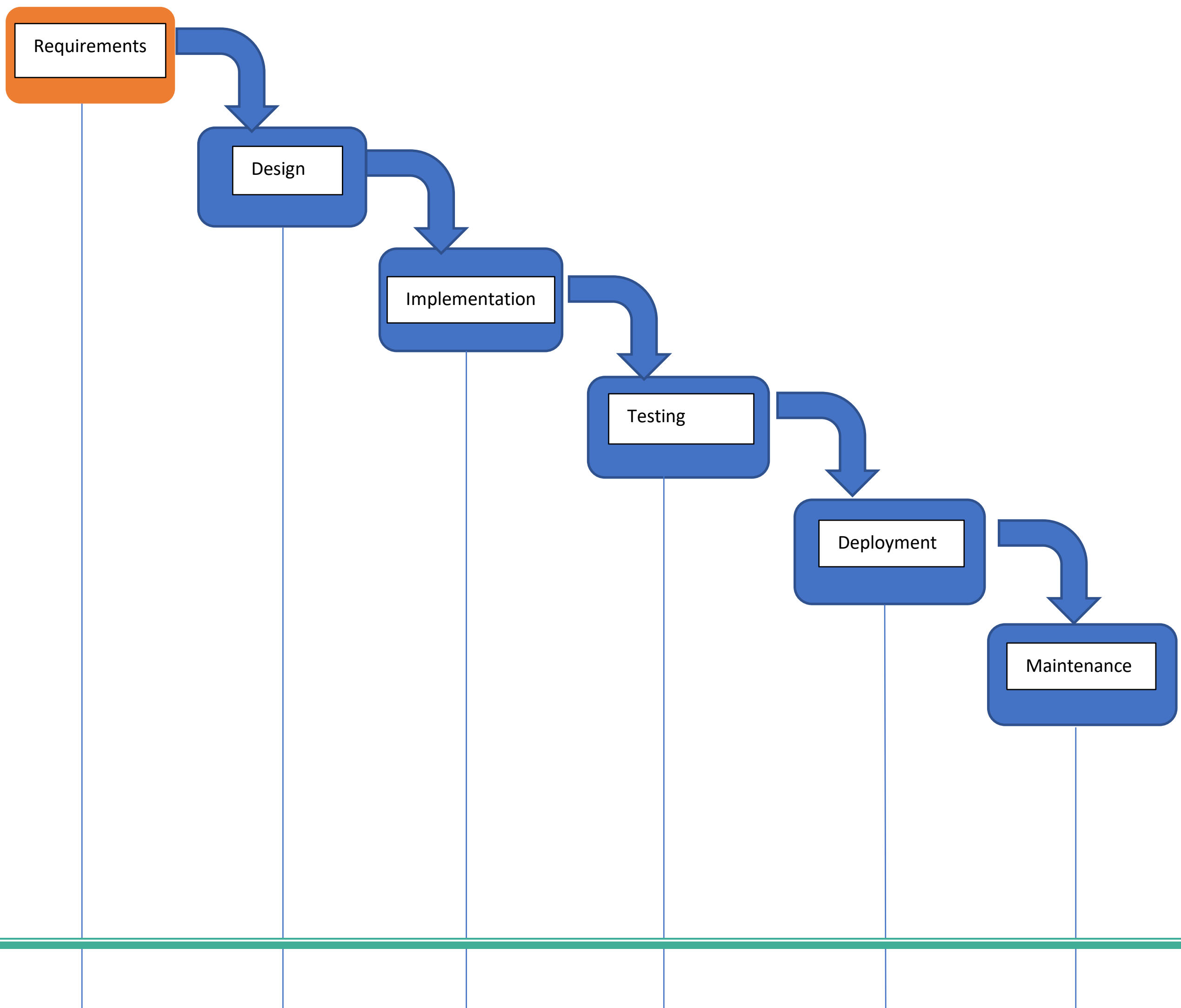
Methodology adapted for Project:

Model Picked:

We picked Waterfall Model as our Model for our project because waterfall model is very simple to implement and doesn’t require much experienced team. It is perfect for these small scale projects.

One milestone of waterfall is that it doesn’t require much effort to track process and to arrange tasks. We can easily document the process using this model.

Flow chart of Waterfall:



Requirements:

Tool used & Why?:

Python- For programming base

Pip libraries-to perform different functionality

Sublime test editor-to write code

VS code-to write code for different expectations

Terminal - To run code

Google collab / Jupyter notebook – Instant performance check

Google chrome - For research work and as pdf opener

Libraries used:

Cv2/Computer vision/opencv – For cropping images and Applying effects

Tkinter -For GUI

Numpy- For array manipulation when image in array form

Sys – To handle interaction with system

Img2pdf – To save results

Pillow – To apply Different operation on images

Tesseract – To perform OCR

Pathlib – To consider directory paths

Tools Description:

Tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Code eg:

```
#!/usr/bin/python

import Tkinter
top = Tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

Attributes:
Height
Width
Anchor
Fonts
Bg
Fg
Cursors

Widgets:
Button
Canvas
Checkbox
Entry
Frame
Label
Listbox
Text
etc

Geometry:
Pack()
Grid()
Place()

PIL(Pillow):

Python pillow package can be used for creating thumbnails, converting from one format to another and print images, etc.

You can display images using Tk PhotoImage, BitmapImage and Windows DIB interface, which can be used with PythonWin and other Windows-based toolkits and many other Graphical User Interface (GUI) toolkits.

For debugging purposes, there is a show () method to save the image to disk which calls the external display utility.

The Pillow library contains all the basic image processing functionality. You can do image resizing, rotation and transformation.

Pillow module allows you to pull some statistics data out of image using histogram method, which later can be used for statistical analysis and automatic contrast enhancement.

Main use:

Display image

Save image

Rotate image

Resize image

Open image

CV2:

openCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Computer Vision overlaps significantly with the following fields −

- **Image Processing** − It focuses on image manipulation.
- **Pattern Recognition** − It explains various techniques to classify patterns.

- **Photogrammetry** − It is concerned with obtaining accurate measurements from images.

Features of OpenCV Library

Using OpenCV library, you can −

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyse the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

Numpy:

NumPy is a Python library used for working with arrays. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

Sublime Text Editor:

Sublime text is a very efficient and programming oriented text editor.

Tesseract:

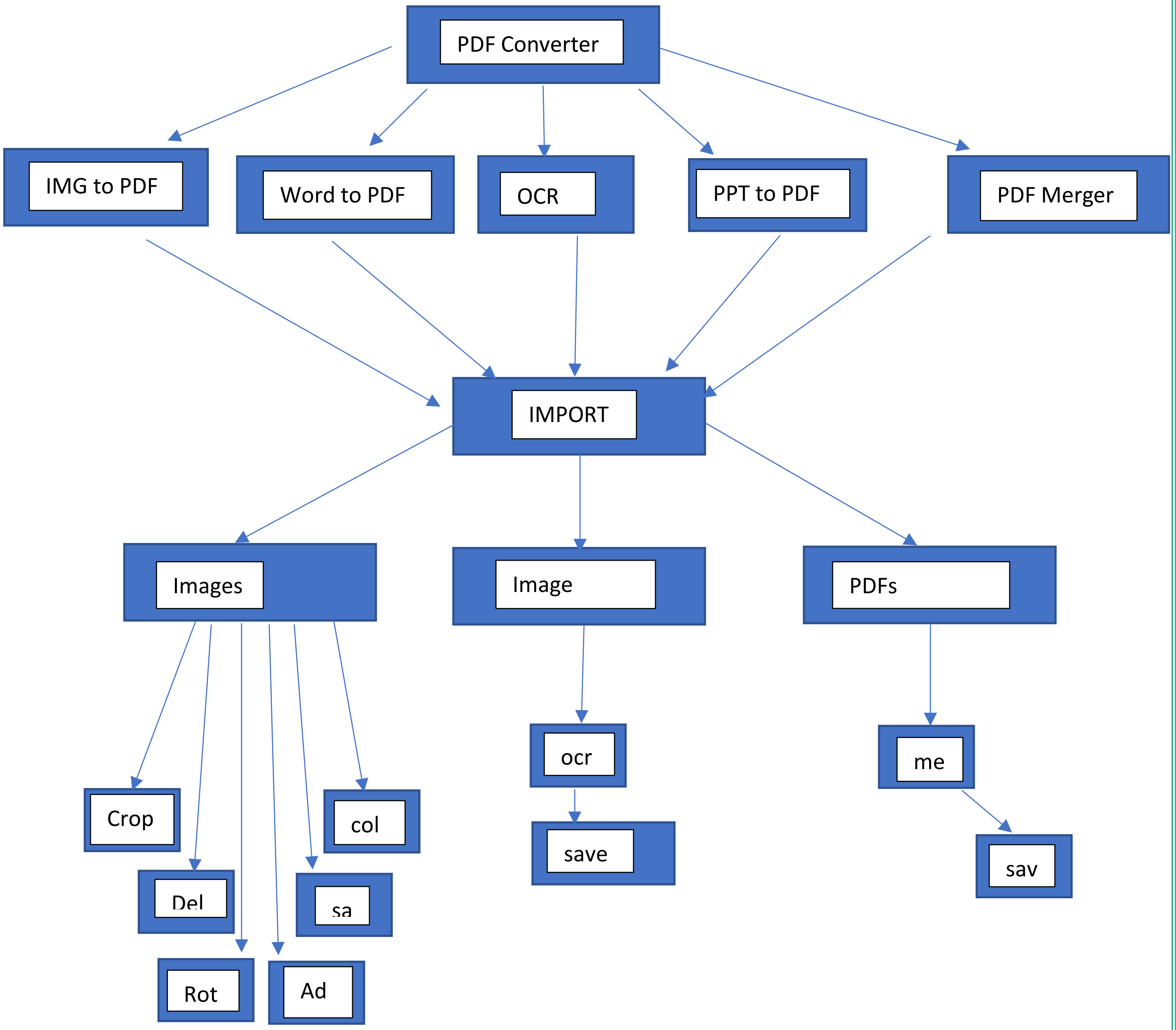
Tesseract is an optical character recognition engine with open-source code, this is the most popular and qualitative OCR-library. OCR uses artificial intelligence for text search and its recognition on images.

Tesseract is finding templates in pixels, letters, words and sentences. It uses two-step approach that calls adaptive recognition. It requires one data stage for character recognition, then the second stage to fulfil any letters, it wasn't insured in, by letters that can match the word or sentence context.

Design:

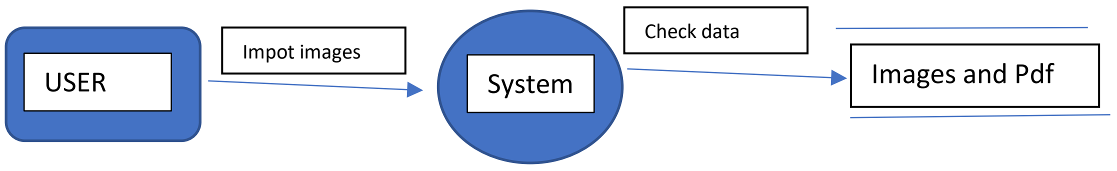
System structure is described in flow chart and DFD's.

Block diagram:

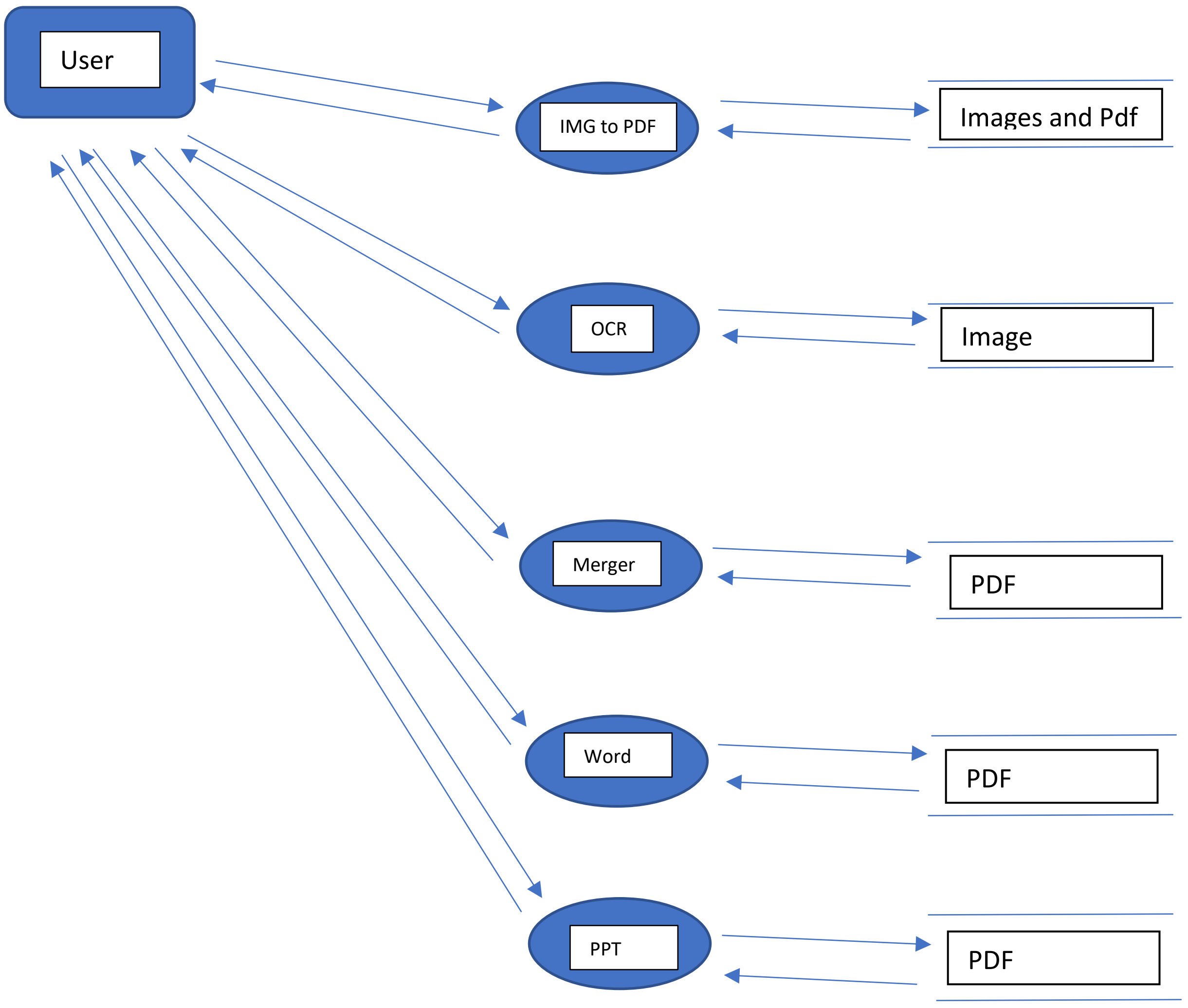


DFD’s:

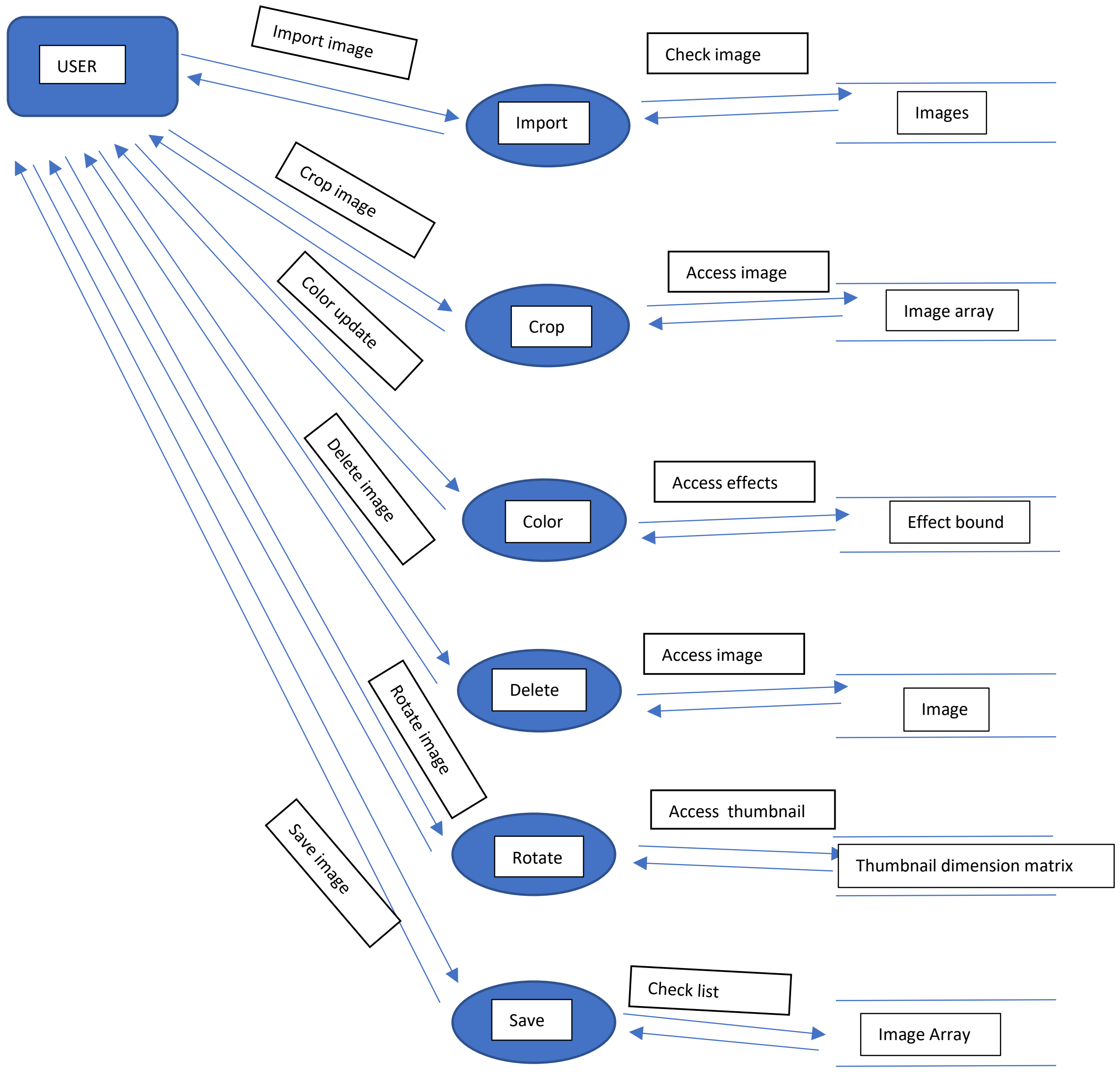
0 level Dfd:



1 level Dfd:



2 level DFD:


USER
Import image
Crop image
Color update
Delete image
Rotate image
Save image
Import
Crop
Color
Delete
Rotate
Save
Check image
Access image
Access effects
Access image
Access thumbnail
Check list
Images
Image array
Effect bound
Image
Thumbnail dimension matrix
Image Array

Implementation/Functions Of the Application:

The app has various functions like cropping , color, delete, add file, import, save, rotate etc.

Crop function:

Crop code uses 2 libraries PIL(pillow) and cv2. Cv2 offers to select 4 points in image to crop out inscribed area by four points and then PIL performs image conversion from different forms and put back image in list.

Selection of 4 points is in Z format. Pressing S saves the image . Pressing E exits the CV2 viewport.

Prev Button(<<):

Previous Button is a feature to move between images And this particular button brings the previous image in viewer.

Next Button(>>):

Next Button is a feature to move between images And this particular button brings the next image in viewer.

Import Button:

Import button is used to import images and all other supported files from directory to our application.

Add Files:

Add Flies is used to add more images or all other supported files in application.

Color:

Color button provides effect to images to give more readable output.

Rotate:

Rotate Button is used to rotate a image. It is implemented as 90 degree counterclock-wise.

Save pdf/Convert:

This button saves the images imported in application in pdf format.

Delete :

This button provides the functionality of deleting the current image in viewer.

Dropbox and Insert:

It inserts file at selected position in dropbox.

Dropbox and Delete:

It deletes the file selected in dropbox

Code:

```
#import Libraries

import os

from tkinter import *

import tkinter as tk

from tkinter import ttk

from tkinter import filedialog

from PIL import ImageTk,Image

import cv2 as cv

import sys

import numpy as np

import img2pdf

from tkinter.scrolledtext import ScrolledText   #scrollable text box

import PyPDF2  #pdf merging
import win32com.client # for Word and ppd to pdf
import pytesseract    #OCR
# initiate tkinter
root=tk.Tk()
root.title("PDF Converter")
#geometry for window
root.geometry('750x770') ##(w x h)
root.minsize(750,770)
root.maxsize(750,770)
root.configure(bg='#26A65B',bd=6)

#Image to pdf converting
def IMGTOPDF():
    global f3
    if f3:
        f3.destroy()
    f3=Frame(root,bg="green",bd=4)
    f3.grid(row=1,column=0,rowspan=10,columnspan=9)

    list1=[]# it stores link of import item
    thumb=[]# it store rezized thumbnail of pictures
    listcrop=[]# it stores the product images
    listpdf=[]#final list of image before pdf conversion

    n=0 #no of images
    i=0 #count for current image in view
    # function for thumbnail generation
    def tumbnail(imgx):

            imgx = imgx.resize((600,700), Image.ANTIALIAS)
            photoImg =  ImageTk.PhotoImage(imgx)
            return photoImg

    #Import function
    def insertfun():
          global f3
          nonlocal list1
          nonlocal thumb
          nonlocal n
          nonlocal i
          root.withdraw()
          files=filedialog.askopenfilenames(title="insert",filetypes=(("png files","*.png"),("allfiles","*.*")))



          filez = root.tk.splitlist(files)
          root.update()
          root.deiconify()

          for filex in filez:
              list1.append(filex)
              img = Image.open(filex)
              imggg=tumbnail(img)
              listcrop.append(img)
              thumb.append(imggg)

          i=0
          n=len(thumb)
          #redefining canvas
          canva=Canvas(f3,height=700,width=650,bg="lightsteelblue2")
          canva.grid(row=1,column=1,columnspan=6,rowspan=9)
          imago=canva.create_image(0,0,anchor=NW,image=thumb[i])

    #forward button function
    def forw():
          nonlocal list1
          nonlocal thumb
          nonlocal i
          global f3
          if i<len(thumb)-1:
              i=i+1
              canva=Canvas(f3,height=700,width=650,bg="lightsteelblue2")
              canva.grid(row=1,column=1,columnspan=6,rowspan=9)
              imago=canva.create_image(0,0,anchor=NW,image=thumb[i])
              l3 = Button(f3,text="<<",state=NORMAL,command=backw)
              l3.grid(row=5,column=0)
          else:
              l4 = Button(f3,text=">>",state=DISABLED)
              l4.grid(row=5,column=7)
```

```
#Backward button function
def backw():
    nonlocal list1
    nonlocal thumb
    nonlocal i
    global f3
    if i>=0 :
        i=i-1
        canva=Canvas(f3,height=700,width=650,bg="lightsteelblue2")
        canva.grid(row=1,column=1,columnspan=6,rowspan=9)
        imago=canva.create_image(0,0,anchor=NW,image=thumb[i])
        l4 = Button(f3,text=">>",state=NORMAL,command=forw)
        l4.grid(row=5,column=7)
    else:
        l3 = Button(f3,text="<<",state=DISABLED)
        l3.grid(row=5,column=0)



#delete function to delete at count image
def deletee():
        nonlocal list1
        nonlocal thumb
        nonlocal i
        nonlocal n
        global f3
        list1.pop(i)
        thumb.pop(i)
        listcrop.pop(i)
        n=len(thumb)
        if i==n-1:
            i-=1
        if i==0:
            if thumb.empty():

                canva=Canvas(f3,height=700,width=650,bg="lightsteelblue2")
                canva.grid(row=1,column=1,columnspan=6,rowspan=9)
            else:
           i
        canva=Canvas(f3,height=700,width=650,bg="lightsteelblue2")
        canva.grid(row=1,column=1,columnspan=6,rowspan=9)
        imago=canva.create_image(0,0,anchor=NW,image=thumb[i])

#to crop image (main function for project)
def cropp():
        nonlocal list1
        nonlocal i
        nonlocal thumb
        nonlocal n
        nonlocal listcrop
        nonlocal listpdf
        global f3
        listxx=[]
        img=cv.imread(cv.samples.findFile(list1[i]))

        if img is None:
            sys.exit("Could not read the image")

        screen_res = 1280, 720
        scale_width = screen_res[0] / img.shape[1]
        scale_height = screen_res[1] / img.shape[0]
        scale = min(scale_width, scale_height)
            #resized window width and height
        window_width = int(img.shape[1] * scale)
        window_height = int(img.shape[0] * scale)

        cv.namedWindow('Image Display',cv.WINDOW_NORMAL)
        cv.resizeWindow('Image Display',window_width,window_height)
        cv.imshow("Image Display",img)
        def click_event(event, x, y, flags, params):
             if len(listxx)<=3:
                        # checking for left mouse clicks
                 if event == cv.EVENT_LBUTTONDOWN:
                     listxx.append([x,y])
                           # displaying the coordinates
                           # on the Shell
```

```
            cv.imshow('Image Display', img)
    if len(listxx)==4:


          pts = np.array(listxx, np.int32)
          while True:
               pts=np.float32(pts)


               pts2 = np.float32([[0, 0], [959, 0], [0, 1279], [959, 1279]])


               # Apply Perspective Transform Algorithm
               matrix = cv.getPerspectiveTransform(pts, pts2)
               result = cv.warpPerspective(img, matrix, (img.shape[1], img.shape[0]))


               # Wrap the transformed image
               cv.namedWindow('Image',cv.WINDOW_NORMAL)
               cv.resizeWindow('Image',window_width,window_height)


               cv.imshow('Image', result) # Transformed Capture
               result= cv.cvtColor(result, cv.COLOR_BGR2RGB)
               im_pil = Image.fromarray(result)
               listcrop[i]=im_pil
               thumb[i]=tumbnail(im_pil)
               #redefining canvas
               canva=Canvas(f3,height=700,width=650,bg="lightsteelblue2")
               canva.grid(row=1,column=1,columnspan=6,rowspan=9)
               imago=canva.create_image(0,0,anchor=NW,image=thumb[i])
               #waiting for key input to end terminal
               zz=cv.waitKey(0)
               if zz==ord('e'):
                    cv.destroyAllWindows()
                    break
               if zz==ord('s'):
                    listcrop[i]=result
                    cv.destroyAllWindows()
                    break



#to set terminal in loopback
cv.setMouseCallback("Image Display", click_event)
```

```
#function to rotate image
def rotatee():
        nonlocal i
        nonlocal listcrop
        nonlocal thumb
        nonlocal n
        global f3
        imgho=listcrop[i]
        imgho=imgho.rotate(90)
        listcrop[i]=imgho
        thumb[i]=tumbnail(imgho)
        canva=Canvas(f3,height=700,width=650,bg="lightsteelblue2")
        canva.grid(row=1,column=1,columnspan=6,rowspan=9)
        imago=canva.create_image(0,0,anchor=NW,image=thumb[i])



#function to save image
def makepdf():
     nonlocal listcrop
     nonlocal list1

     if len(thumb)!=0:
         lixrt=listcrop[1:]



         im1 = listcrop[0].convert('RGB')
         for xx in range(n-1):
             lixrt[xx]=lixrt[xx].convert('RGB')
         file=filedialog.asksaveasfilename()

         im1.save(file+".pdf","PDF",save_all=True, append_images=lixrt)
         print("saved")
         # #pdf_bytes = img2pdf.convert(im1,filename)
```

```
            # im1=Image.open(lixrt[0].filename())
            # im1 = im1.convert('RGB')
            # im1.save(r'C:/Users/a/OneDrive/Pictures/l.pdf',"PDF")
            # print("saved")
    #all Gui widgets for IMGtoPDF
    l3 = Button(f3,text="<<",command=backw)
    l3.grid(row=5,column=0,sticky=N)
    l4 = Button(f3,text=">>",command=forw)
    l4.grid(row=5,column=7)
    canva=Canvas(f3,height=700,width=584,bg="lightsteelblue2")
    canva.grid(row=1,column=1,columnspan=6,rowspan=9)
    l5 = Button(f3,text="Import",command=insertfun,width=10)
    l5.grid(row=1,column=8)
    l6 = Button(f3,text="Add Item",width=10)
    l6.grid(row=2,column=8)
    l11 = Button(f3,text="Save PDF",command=makepdf,width=10)
    l11.grid(row=10,column=8)
    l7 = Button(f3,text="crop",command=cropp,width=10)
    l7.grid(row=10,column=1)
    l8 = Button(f3,text="Rotate",command=rotatee,width=10)
    l8.grid(row=10,column=2)
    l10 = Button(f3,text="Color",width=10)
    l10.grid(row=10,column=3)
    l9 = Button(f3,text="Delete",command=deletee,width=10)
    l9.grid(row=10,column=4)





# Tesseract OCR library
def OCR():
    global f3
    if f3:
        f3.destroy()
    f3=Frame(root,bg="green",bd=8)
    f3.grid(row=1,column=0,rowspan=10,columnspan=9)

    file=""
```

```
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract'

frame=Frame(f3,width=600,height=300)
frame.grid(row=1,column=0,columnspan=8,rowspan=5)

canvas=Canvas(frame,bg='#FFFFFF',width=584,height=350,scrollregion=(0,0,600,600))

hbar=Scrollbar(frame,orient=HORIZONTAL)
hbar.pack(side=BOTTOM,fill=X)
hbar.config(command=canvas.xview)
vbar=Scrollbar(frame,orient=VERTICAL)
vbar.pack(side=RIGHT,fill=Y)
vbar.config(command=canvas.yview)
frame1=Frame(f3,width=300,height=300)
frame1.grid(row=6,column=0,columnspan=8,rowspan=4)

text_area=ScrolledText(frame1,height =20,width=73,font=("TkFixedFont",11))
text_area.pack(side=LEFT,anchor=NW,expand=True,fill=BOTH)


canvas.config(width=584,height=300)
canvas.config(xscrollcommand=hbar.set, yscrollcommand=vbar.set)
canvas.pack(side=LEFT,expand=True,fill=BOTH)

def impo():
    nonlocal file
    file=""
    root.update()
    root.withdraw()
    file=filedialog.askopenfilename()
    root.update()
    root.deiconify()
    root.photoImg =  ImageTk.PhotoImage(file="C:\\Users\\ajayk\\Downloads\\x.png")
    imo=canvas.create_image(0,0, anchor=NW, image=root.photoImg)
def conv():
    nonlocal file
    if file!="":
        image = cv.imread(file)
        gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
```

```
            # check to see if we should apply thresholding to preprocess the
            # image

            gray = cv.threshold(gray, 0, 255,
                cv.THRESH_BINARY | cv.THRESH_OTSU)[1]
            # make a check to see if median blurring should be done to remove
            # noise

            gray1 = cv.medianBlur(gray, 3)
            # write the grayscale image to disk as a temporary file so we can
            # apply OCR to it
            filename = "{}.png".format(os.getpid())
            cv.imwrite(filename, gray)
            # load the image as a PIL/Pillow image, apply OCR, and then delete
            # the temporary file
            text = pytesseract.image_to_string(Image.open(filename))
            os.remove(filename)
            print(text)
            text_area.insert(INSERT,"maxo")
            text_area.insert(END, text)

            print( text_area.get(1.0, END) )
            # show the output images
            cv.imshow("Image", image)
            cv.imshow("Output", gray)
            cv.waitKey(0)

        ll1=Button(f3,text="IMPORT IMG",command=impo,width=10)
        ll1.grid(row=1,column=8)
        #ll2=Button(root,text="ADD File",width=10)
        #ll1.grid(row=2,column=8)
        ll3=Button(f3,text="OCR",command=conv,width=10)
        ll3.grid(row=10,column=8)
        #ll4=Button(root,text="IMPORT IMG",command=,width=10)
        #ll4.grid(row=9,column=8)

    #def CLEAR():

    def PPTtoPDF():
```

```
global f3
if f3:

   f3.destroy()
f3=Frame(root,bg="green",bd=8)
f3.grid(row=1,column=0,rowspan=10,columnspan=9)

wdFormatPDF = 32
height=700
width=600
canvas=Canvas(f3,height=height,width=width,bg="lightsteelblue2")
canvas.grid(row=1,column=1,columnspan=6,rowspan=9)
listtt=[]
listtt1=[]
options=["None"]
text_area=ScrolledText(f3,height =41,width=73,font=("TkFixedFont",11))
text_area.grid(row=1,column=1,rowspan=9,columnspan =7)
def importtt():
    listtt.clear()
    listtt1.clear()
    root.update()
    root.withdraw()
    files=filedialog.askopenfilenames(title="insert",filetypes=(("ppt files","*.pptx"),("allfiles","*.*")))
    root.update()
    root.deiconify()
    print("succes")

    for imp in files:


         imp=imp.replace("/","\\")
         name=imp.rsplit('\\')[-1].rsplit(".")[0]
         listtt.append(imp)
         listtt1.append(name)
    print(listtt,listtt1)
    monoo()
def addfile():
    root.update()
    root.withdraw()
    files=filedialog.askopenfilenames(title="insert",filetypes=(("ppt files","*.pptx"),("allfiles","*.*")))
    root.update()
    root.deiconify()


    for imp in files:



         imp=imp.replace("/","\\")
         name=imp.rsplit('\\')[-1].rsplit(".")[0]
         listtt.append(imp)
         listtt1.append(name)
    monoo()
def monoo():
    nonlocal options
    nonlocal listtt1
    options=["None"]
    for i in range(1,len(listtt1)+1):
        options.append(i)
    dropp.set("None")
    inse.set("None")
    drop=OptionMenu(f3,dropp,*options)
    ins=OptionMenu(f3,inse,*options)
    ins.grid(row=3,column=8)
    drop.grid(row=6,column=8)
    more()

def more():
    text_area=ScrolledText(f3,height =41,width=73,font=("TkFixedFont",11))
    text_area.grid(row=1,column=1,rowspan=9,columnspan =7)
    for i in listtt1:
       text_area.insert(INSERT,i+"\n")




def inser():
    if inse.get()!="None":
         ji=int(inse.get())
         files=filedialog.askopenfilename()
```

```
          if files!="":
             i=files.replace("/","\\")
             name=i.rsplit('\\')[-1].rsplit(".")[0]

             listtt.insert(ji-1,i)
             listtt1.insert(ji-1,name)
             print(listtt,listtt1)
             monoo()

def dele():
    nonlocal listtt
    nonlocal listtt1
    if dropp.get()!="None":
        joo=int(dropp.get())

        listtt.pop(joo-1)
        listtt1.pop(joo-1)
    monoo()



def conv():
    root.update()
    root.withdraw()
    m=filedialog.askdirectory()
    m=m.replace("/","\\")
    root.update()
    root.deiconify()
    power = win32com.client.Dispatch('Powerpoint.Application')

    for jo in range(len(listtt)):
        powerpoint = power.Presentations.Open(listtt[jo])
        out_file=listtt1[jo]
        print(out_file)

        powerpoint.SaveAs(m+out_file, FileFormat=wdFormatPDF)
        powerpoint.Close()

    power.Quit()
    lll1=Button(f3,text="IMPORT PPT",width=10,command=importtt).grid(row=1,column=8)
    lll2=Button(f3,text="ADD FILE",width=10,command=addfile).grid(row=2,column=8)
    lll3=Button(f3,text="CONVERT",width=10,command=conv).grid(row=10,column=8)
    llll4=Button(f3,text="DEL",width=10,command=dele).grid(row=7,column=8)
    llll5=Button(f3,text="Insert",width=10,command=inser).grid(row=4,column=8)
    dropp=StringVar()
    dropp.set("None")
    inse=StringVar()
    inse.set("None")
    ins=OptionMenu(f3,inse,*options)
    ins.grid(row=3,column=8)
    drop=OptionMenu(f3,dropp,*options)
    drop.grid(row=6,column=8)
    #lll1=Button(root,text="IMPORT IMG",width=10)



def DOCtoPDF():
    global f3
    if f3:
      f3.destroy()
    f3=Frame(root,bg="green",bd=8)
    f3.grid(row=1,column=0,rowspan=10,columnspan=9)
    wdFormatPDF = 17
    height=700
    width=600
    canvas=Canvas(f3,height=height,width=width,bg="lightsteelblue2")
    canvas.grid(row=1,column=1,columnspan=6,rowspan=9)
    text_area=ScrolledText(f3,height =41,width=73,font=("TkFixedFont",11))
    text_area.grid(row=1,column=1,rowspan=9,columnspan =7)
    lis=[]
    li=[]
    options=["None"]

    def importtt():


        lis.clear()
        li.clear()
    root.update()

    root.withdraw()

    files=filedialog.askopenfilenames()


    root.update()

    root.deiconify()

    for i in files:



        i=i.replace("/","\\")

        name=i.rsplit('\\')[-1].rsplit(".")[0]

        lis.append(i)

        li.append(name)

    monoo()


    print("import",lis,li)

    #maxo()

def addfile():

    root.update()

    root.withdraw()

    files=filedialog.askopenfilenames()


    root.update()

    root.deiconify()

    for i in files:



        i=i.replace("/","\\")

        name=i.rsplit('\\')[-1].rsplit(".")[0]

        lis.append(i)

        li.append(name)

    monoo()

    print("add file",lis,li)

def more():

   global f3

   text_area=ScrolledText(f3,height =41,width=73,font=("TkFixedFont",11))

   text_area.grid(row=1,column=1,rowspan=9,columnspan =7)

   for i in li:

       text_area.insert(INSERT,i+"\n")
```

```
# def maxo():

#      imo=filedialog.askopenfilename()
#      immo=Image.open(imo)
#      immmo=immo.resize((50,50), Image.ANTIALIAS)
#      print(os.getcwd())
#      global canva
#      img=ImageTk.PhotoImage(immo)
#      print(imo,immo,immmo,img)


#    imago=canva.create_image(0,0,image=img,anchor=NW)

def monoo():
    nonlocal li
    nonlocal options
    global f3
    options=["None"]
    for i in range(1,len(li)+1):
        options.append(i)
    dropp.set("None")
    inse.set("None")
    drop=OptionMenu(f3,dropp,*options)
    ins=OptionMenu(f3,inse,*options)
    ins.grid(row=3,column=8)
    drop.grid(row=6,column=8)
    print("monno",lis,li)
    more()
def inser():
    if inse.get()!="None":
        ji=int(inse.get())
        files=filedialog.askopenfilename()
        if files!="":
            i=files.replace("/","\\")
            name=i.rsplit('\\')[-1].rsplit(".")[0]

            lis.insert(ji-1,i)
            li.insert(ji-1,name)
```

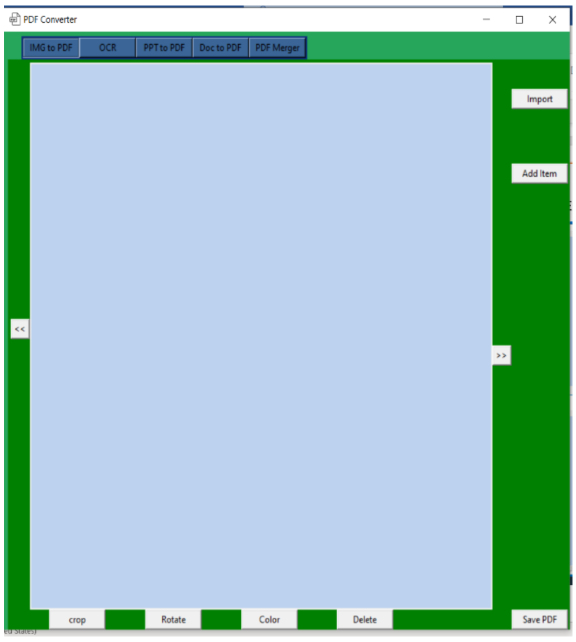
```
              monoo()
              print("inser",lis,li)
def dele():
     nonlocal lis
     nonlocal li
     if dropp.get()!="None":
        joo=int(dropp.get())

        lis.pop(joo-1)
        li.pop(joo-1)
     monoo()
     print("dele",lis,li)

def cont():
     root.update()
     root.withdraw()
     m=filedialog.askdirectory()
     m=m.replace("/","\\")
     root.update()
     root.deiconify()

     word = win32com.client.Dispatch('Word.Application')
     for i in range(len(lis)):
          doc = word.Documents.Open(lis[i])
          out_file=m+li[i]
          print(out_file)
          doc.SaveAs(out_file, FileFormat=wdFormatPDF)
          doc.Close()
     word.Quit()
llll1=Button(f3,text="IMPORT DOC",width=10,command=importtt).grid(row=1,column=8)
llll2=Button(f3,text="ADD FILE",width=10,command=addfile).grid(row=2,column=8)
llll3=Button(f3,text="CONVERT",width=10,command=cont).grid(row=10,column=8)
llll4=Button(f3,text="DEL",width=10,command=dele).grid(row=7,column=8)
llll5=Button(f3,text="Insert",width=10,command=inser).grid(row=4,column=8)
dropp=StringVar()
dropp.set("None")
inse=StringVar()
inse.set("None")
ins=OptionMenu(f3,inse,*options)
```

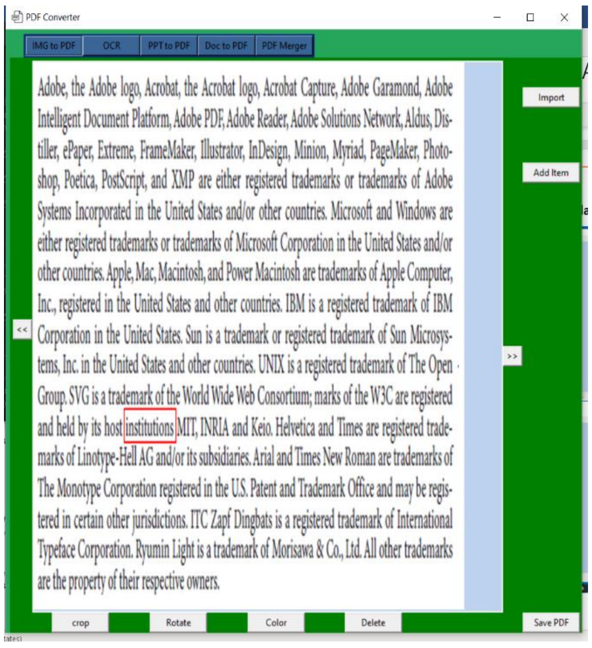
```
    ins.grid(row=3,column=8)

    drop=OptionMenu(f3,dropp,*options)

    drop.grid(row=6,column=8)



def PDFMERGER():

    global f3

    if f3:

      f3.destroy()

    f3=Frame(root,bg="#3A6596",bd=8)

    f3.grid(row=1,column=0,rowspan=10,columnspan=9)

    pdf2merge=[]

    pdfWriter = PyPDF2.PdfFileWriter()

    height=700

    width=600

    canvas=Canvas(f3,height=height,width=width,bg="lightsteelblue2")

    canvas.grid(row=1,column=1,columnspan=6,rowspan=9)

    text_area=ScrolledText(f3,height =41,width=76,font=("TkFixedFont",11))

    text_area.grid(row=1,column=1,rowspan=9,columnspan =7)

    options=["None"]

    def addo():

        pdf2merge.clear()

        root.update()

        root.withdraw()                    # asks users where the PDFs are

        fileso = filedialog.askopenfilenames()

        root.update()

        root.deiconify()

        for izo in fileso:


          pdf2merge.append(izo)


        monoo()


    def addfile():

        root.update()

        root.withdraw()                    # asks users where the PDFs are

        fileso = filedialog.askopenfilenames()

        root.update()

        for izo in fileso:
          pdf2merge.append(izo)
      monoo()
def more():
   global f3
   text_area=ScrolledText(f3,height =41,width=73,font=("TkFixedFont",11))
   text_area.grid(row=1,column=1,rowspan=9,columnspan =7)
   for i in pdf2merge:
       text_area.insert(INSERT,i+"\n")



def monoo():
        nonlocal pdf2merge
        nonlocal options
        global f3
        options=["None"]
        for i in range(1,len(pdf2merge)+1):
                 options.append(i)
        dropp.set("None")
        inse.set("None")
        drop=OptionMenu(f3,dropp,*options)
        ins=OptionMenu(f3,inse,*options)
        ins.grid(row=3,column=8)
        drop.grid(row=6,column=8)
        more()
def inser():
        if inse.get()!="None":
              ji=int(inse.get())
              files=filedialog.askopenfilename()
              if files!="":
                  pdf2merge.insert(ji-1,files)


                  monoo()
                  print("inser",lis,li)


def dele():
       nonlocal pdf2merge
       if dropp.get()!="None":
             joo=int(dropp.get())
```

```
            pdf2merge.pop(joo-1)
        monoo()



    # Ask user for the name to save the file as


    def convo():
    # loop through all PDFs
        for filename in pdf2merge:
            # rb for read binary
            pdfFileObj = open(filename, 'rb')
            pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
            # Opening each page of the PDF
            for pageNum in range(pdfReader.numPages):
                pageObj = pdfReader.getPage(pageNum)
                pdfWriter.addPage(pageObj)
        # save PDF to file, wb for write binary
        userfilename = filedialog.asksaveasfilename()
        pdfOutput = open(userfilename + '.pdf', 'wb')
        # Outputting the PDF
        pdfWriter.write(pdfOutput)
        # Closing the PDF writer
        pdfOutput.close()


    lllll1=Button(f3,text="IMPORT PDF",width=10,command=addo).grid(row=1,column=8)
    lllll2=Button(f3,text="ADD FILE",width=10,command=addfile).grid(row=2,column=8)
    lllll3=Button(f3,text="CONVERT",width=10,command=convo).grid(row=10,column=8)
    lllll4=Button(f3,text="DEL",width=10,command=dele).grid(row=7,column=8)
    lllll5=Button(f3,text="Insert",width=10,command=inser).grid(row=4,column=8)
    dropp=StringVar()
    dropp.set("None")
    inse=StringVar()
    inse.set("None")
    ins=OptionMenu(f3,inse,*options)
    ins.grid(row=3,column=8)
    drop=OptionMenu(f3,dropp,*options)
    drop.grid(row=6,column=8)


#all buttons and other widgets initializing
```

```
fr=Frame(root,bg="#0F307D",bd=2)
fr.grid(row=0,column=0,rowspan=1,columnspan=5,sticky=N)

l1 = Button(fr,text="IMG to PDF", fg="black", bg="#3A6596",height=1,width=10,bd=2,relief=GROOVE,command=IMGTOPDF)
l2 = Button(fr,text="OCR", fg="black", bg="#3A6596",width=10,command=OCR)
l14=Button(fr,text="PPT to PDF", fg="black", bg="#3A6596",width=10,command=PPTtoPDF)
l14.grid(row=0,column=3)
l1.grid(row=0,column=1)
l2.grid(row=0,column=2)

l15=Button(fr,text="Doc to PDF", fg="black", bg="#3A6596",width=10,command=DOCtoPDF)
l15.grid(row=0,column=4)
l16=Button(fr,text="PDF Merger", fg="black", bg="#3A6596",width=10,command=PDFMERGER)
l16.grid(row=0,column=5)
#l17=Button(root,text="All to PDF ", fg="black", bg="white",width=10)
#l17.grid(row=0,column=6)

root.iconbitmap('C:/Users/ajayk/Downloads/pdf.ico')




f3=Frame(root,bg="green",bd=2)
f3.grid(row=1,column=0,rowspan=10,columnspan=9)


# to stuck gui in loop
root.mainloop()
```

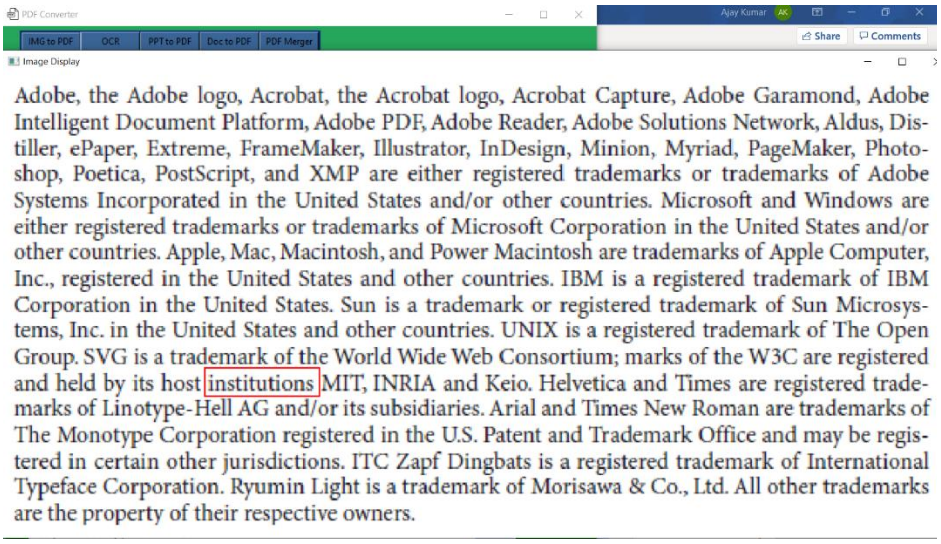
Application Output screens(Explained Usage):





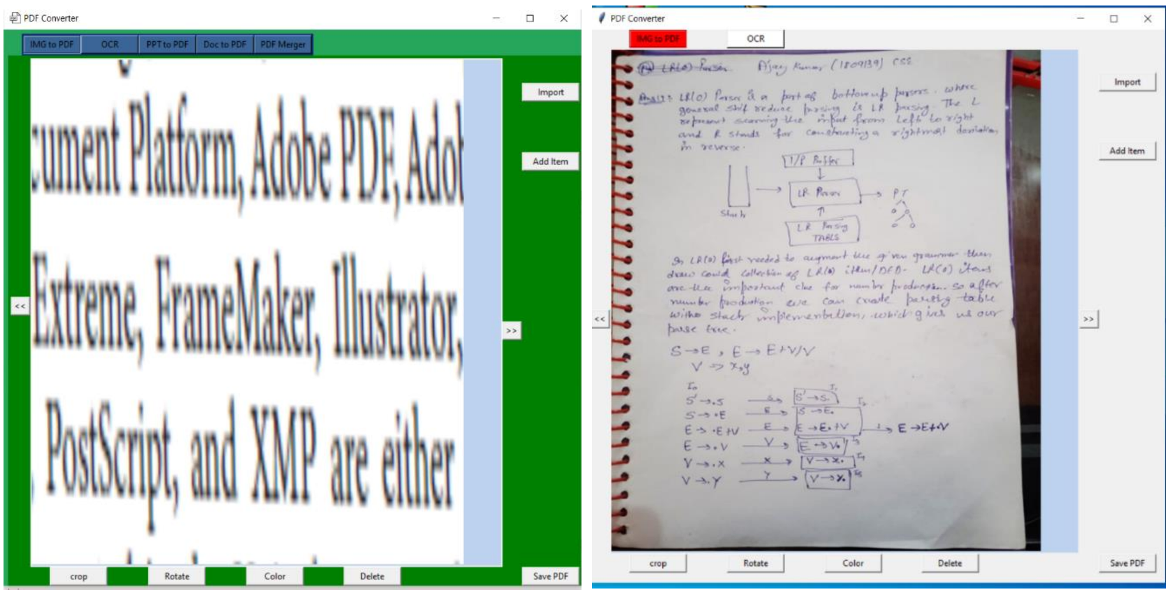
Adobe, the Adobe logo, Acrobat, the Acrobat logo, Acrobat Capture, Adobe Garamond, Adobe Intelligent Document Platform, Adobe PDF, Adobe Reader, Adobe Solutions Network, Aldus, Distiller, ePaper, Extreme, FrameMaker, Illustrator, InDesign, Minion, Myriad, PageMaker, Photoshop, Poetica, PostScript, and XMP are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Apple, Mac, Macintosh, and Power Macintosh are trademarks of Apple Computer, Inc., registered in the United States and other countries. IBM is a registered trademark of IBM Corporation in the United States. Sun is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark of The Open Group. SVG is a trademark of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions MIT, INRIA and Keio. Helvetica and Times are registered trademarks of Linotype-Hell AG and/or its subsidiaries. Arial and Times New Roman are trademarks of The Monotype Corporation registered in the U.S. Patent and Trademark Office and may be registered in certain other jurisdictions. ITC Zapf Dingbats is a registered trademark of International Typeface Corporation. Ryumin Light is a trademark of Morisawa & Co., Ltd. All other trademarks are the property of their respective owners.

Main menu of application and loading image in application



Adobe, the Adobe logo, Acrobat, the Acrobat logo, Acrobat Capture, Adobe Garamond, Adobe Intelligent Document Platform, Adobe PDF, Adobe Reader, Adobe Solutions Network, Aldus, Distiller, ePaper, Extreme, FrameMaker, Illustrator, InDesign, Minion, Myriad, PageMaker, Photoshop, Poetica, PostScript, and XMP are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Apple, Mac, Macintosh, and Power Macintosh are trademarks of Apple Computer, Inc., registered in the United States and other countries. IBM is a registered trademark of IBM Corporation in the United States. Sun is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries. UNIX is a registered trademark of The Open Group. SVG is a trademark of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions MIT, INRIA and Keio. Helvetica and Times are registered trademarks of Linotype-Hell AG and/or its subsidiaries. Arial and Times New Roman are trademarks of The Monotype Corporation registered in the U.S. Patent and Trademark Office and may be registered in certain other jurisdictions. ITC Zapf Dingbats is a registered trademark of International Typeface Corporation. Ryumin Light is a trademark of Morisawa & Co., Ltd. All other trademarks are the property of their respective owners.

DISplaying image



cument Platform, Adobe PDF, Adob
Extreme, FrameMaker, Illustrator,
PostScript, and XMP are either

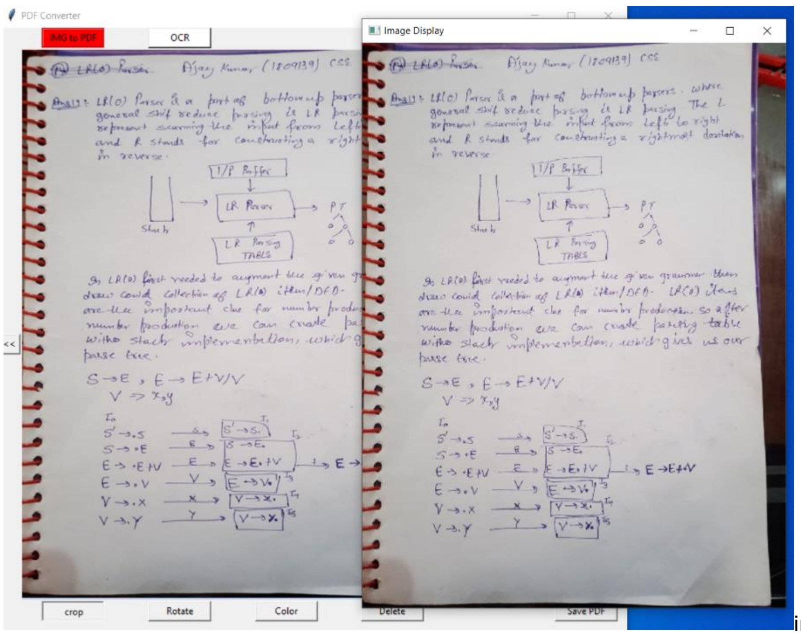
~~LR(0) Parser~~ Ajay Kumar (1809139) CSE

Ans 1:- LR(0) Parser is a part of bottom up parsers. where general shift reduce parsing is LR parsing. The L represent scanning the input from left to right and R stands for constructing a rightmost derivation in reverse.

In LR(0) first needed to augment the given grammar then draw could collection of LR(0) items/DFA. LR(0) items are the important clue for number production. so after number production we can create parsing table with stack implementation, which gives us our parse tree.

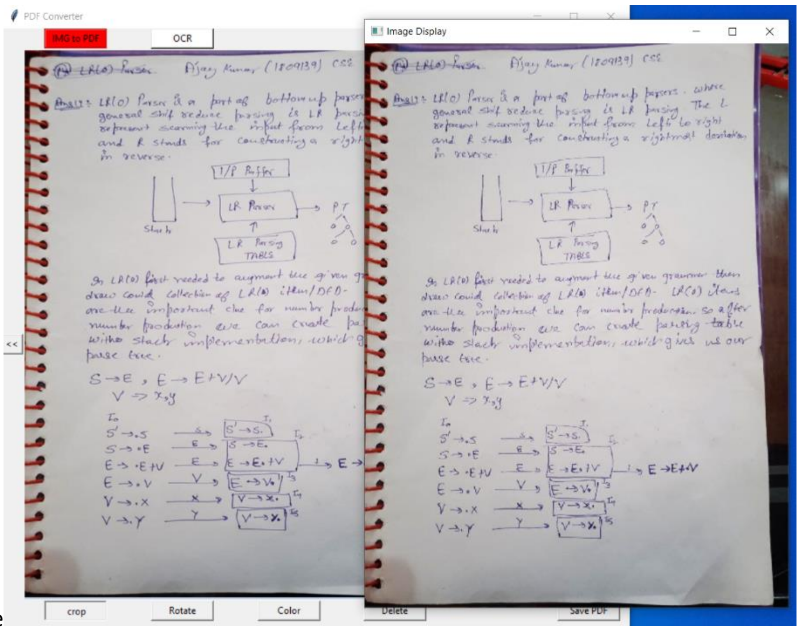
$S \to E$, $E \to E+V/V$
$V \to x, y$

Function of all buttons



ing Import

click on import



~~(b) LR(0) Parser~~ Ajay Kumar (1809139) CSE

Ans 17: LR(0) Parser is a part of bottom up parsers. where general shift reduce parsing is LR parsing. The L represent scanning the input from Left to right and R stands for constructing a rightmost derivation in reverse.

I/P Buffer → LR Parser → PT; Stack → LR Parser; LR Parsing TABLE → LR Parser

In LR(0) first needed to augment the given grammar then draw could collection of LR(0) item/DFD. LR(0) items are the important clue for number production so after number production we can create parsing table with stack implementation, which gives us our parse tree.

$S \to E$, $E \to E+V/V$

$V \Rightarrow x, y$

I_0

$S' \to .S \xrightarrow{S} \boxed{S' \to S.}\ I_1$

$S \to .E \xrightarrow{E} \boxed{S \to E.}\ I_2$

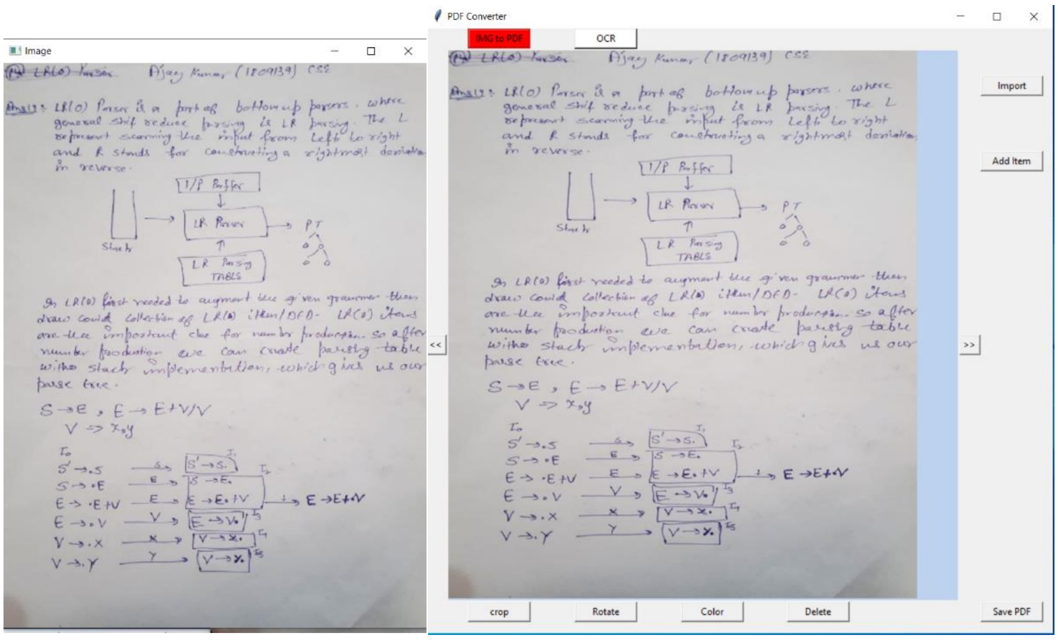
$E \to .E+V \xrightarrow{E} \boxed{E \to E.+V} \xrightarrow{+} E \to E+.V$

$E \to .V \xrightarrow{V} \boxed{E \to V.}\ I_3$

$V \to .x \xrightarrow{x} \boxed{V \to x.}\ I_4$

$V \to .Y \xrightarrow{Y} \boxed{V \to Y.}\ I_5$

select 4 corners of image

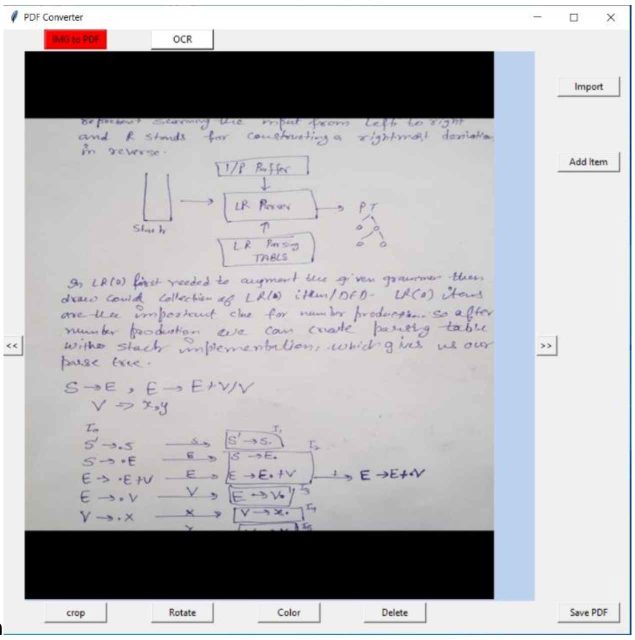


~~(b) LR(0) Parser~~ Ajay Kumar (1809139) CSE

Ans 17: LR(0) Parser is a part of bottom up parsers. where general shift reduce parsing is LR parsing. The L represent scanning the input from Left to right and R stands for constructing a rightmost derivation in reverse.

In LR(0) first needed to augment the given grammar then draw could collection of LR(0) item/DFD. LR(0) items are the important clue for number production so after number production we can create parsing table with stack implementation, which gives us our parse tree.

Crop output



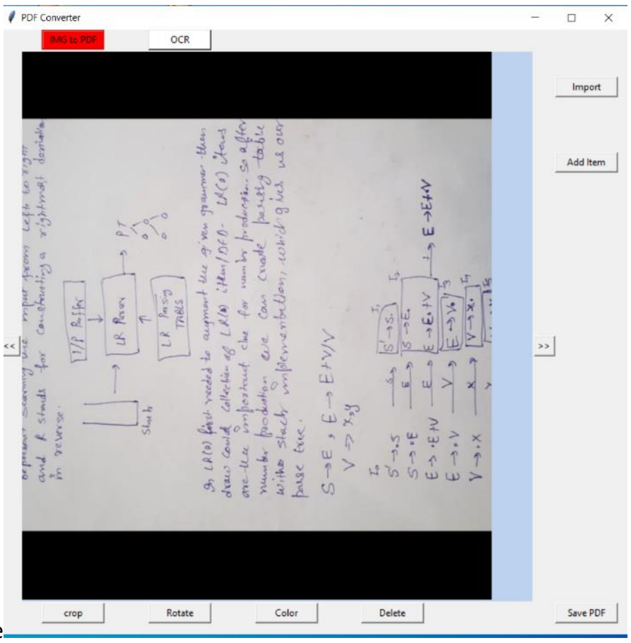
represent scanning the input from left to right and R stands for constructing a rightmost derivation in reverse.

In LR(0) first needed to augment the given grammer then draw could collection of LR(0) item/DFD. LR(0) items are the important clue for number production. so after number production we can create parsing table with the stack implementation, which gives us our parse tree.

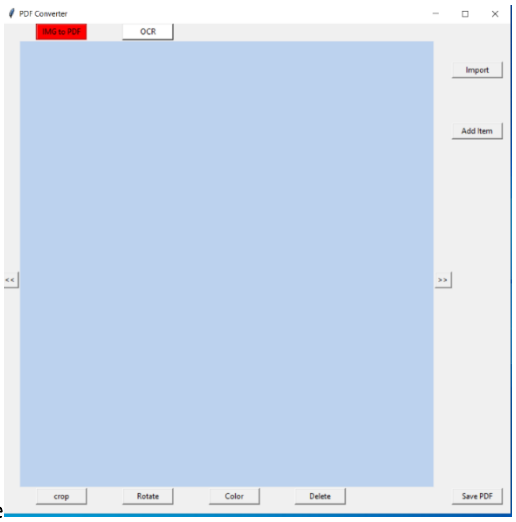
$S \rightarrow E$, $E \rightarrow E+V/V$
$V \Rightarrow x, y$

$$\begin{tikzcd}
I_0 & & \\
S' \rightarrow .S \arrow[r, "S"] & \boxed{S' \rightarrow S.}\ I_1 & \\
S \rightarrow .E \arrow[r, "E"] & \boxed{S \rightarrow E.} & \\
E \rightarrow .E+V \arrow[r, "E"] & \boxed{E \rightarrow E.+V}\ I_2 \arrow[r, "+"] & E \rightarrow E+.V \\
E \rightarrow .V \arrow[r, "V"] & \boxed{E \rightarrow V.}\ I_3 & \\
V \rightarrow .x \arrow[r, "x"] & \boxed{V \rightarrow x.}\ I_4 &
\end{tikzcd}$$

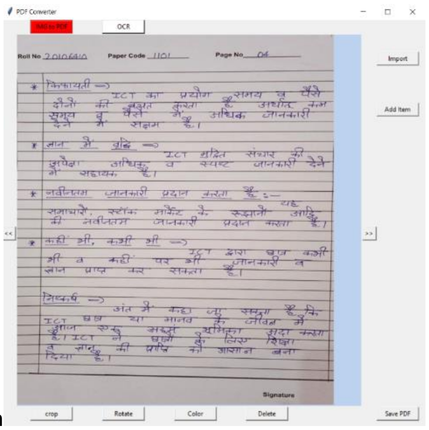
Rotate image by Rotate Button

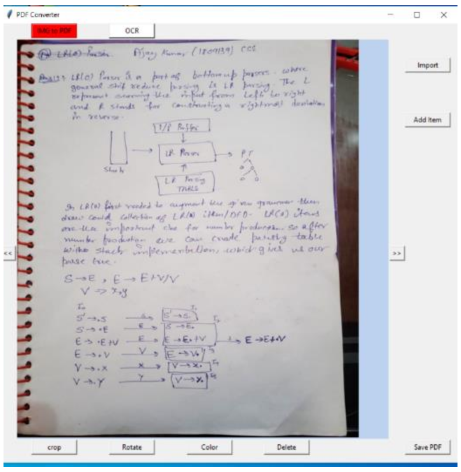


Rotated image



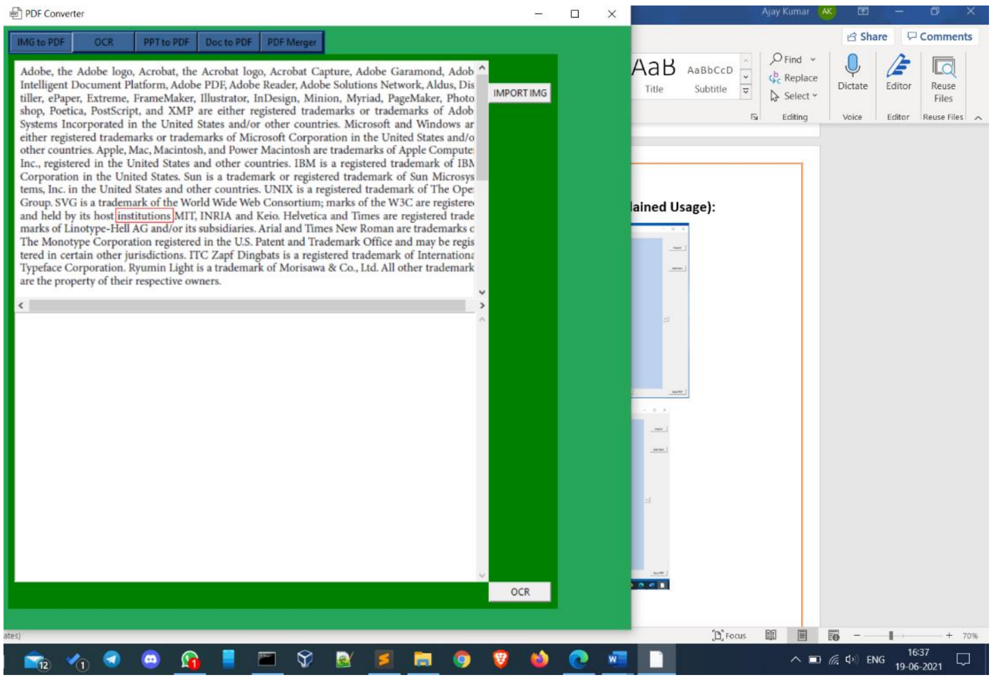
Delete image





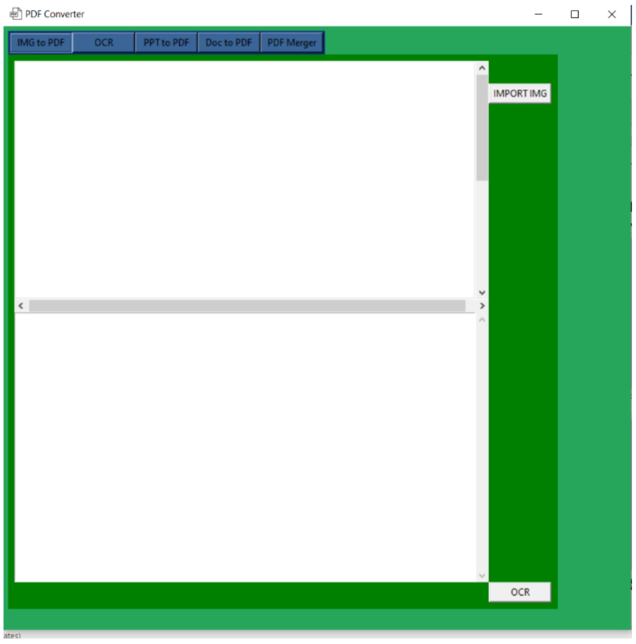
Press Next(>>) Button

Press (<<) Button

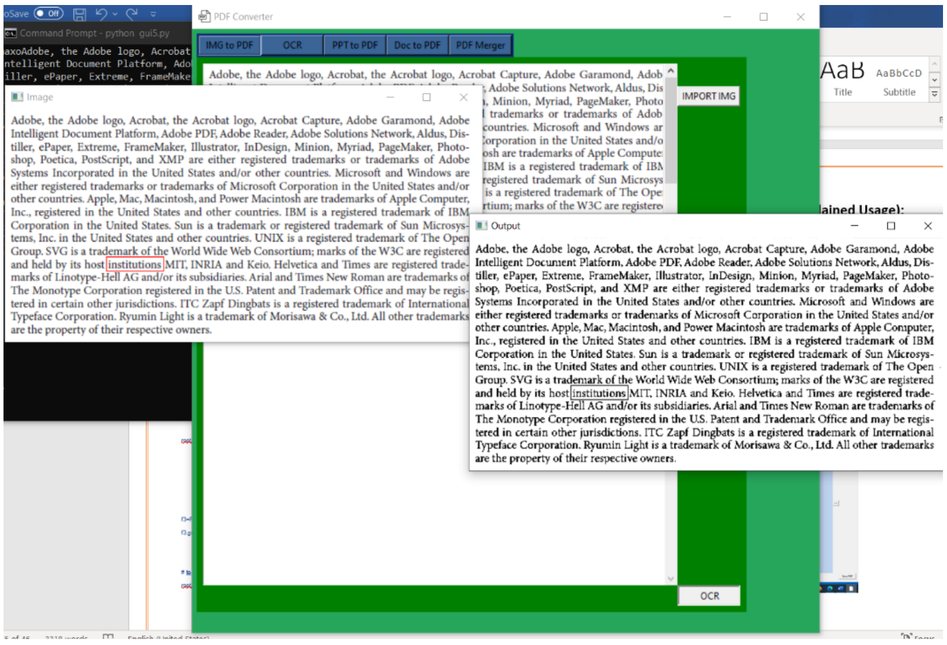


Adobe, the Adobe logo, Acrobat, the Acrobat logo, Acrobat Capture, Adobe Garamond, Adob
Intelligent Document Platform, Adobe PDF, Adobe Reader, Adobe Solutions Network, Aldus, Dis
tiller, ePaper, Extreme, FrameMaker, Illustrator, InDesign, Minion, Myriad, PageMaker, Photo
shop, Poetica, PostScript, and XMP are either registered trademarks or trademarks of Adob
Systems Incorporated in the United States and/or other countries. Microsoft and Windows ar
either registered trademarks or trademarks of Microsoft Corporation in the United States and/o
other countries. Apple, Mac, Macintosh, and Power Macintosh are trademarks of Apple Compute
Inc., registered in the United States and other countries. IBM is a registered trademark of IBM
Corporation in the United States. Sun is a trademark or registered trademark of Sun Microsys
tems, Inc. in the United States and other countries. UNIX is a registered trademark of The Ope
Group. SVG is a trademark of the World Wide Web Consortium; marks of the W3C are registere
and held by its host institutions MIT, INRIA and Keio. Helvetica and Times are registered trade
marks of Linotype-Hell AG and/or its subsidiaries. Arial and Times New Roman are trademarks c
The Monotype Corporation registered in the U.S. Patent and Trademark Office and may be regis
tered in certain other jurisdictions. ITC Zapf Dingbats is a registered trademark of Internationa
Typeface Corporation. Ryumin Light is a trademark of Morisawa & Co., Ltd. All other trademark
are the property of their respective owners.

OCR Inserted image



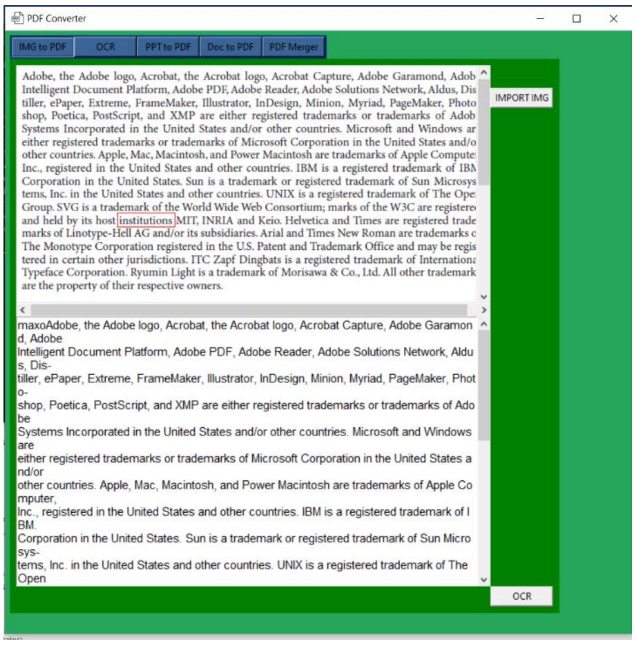
OCR main view



```
Adobe, the Adobe logo, Acrobat, the Acrobat logo, Acrobat Capture, Adobe Garamond, Adobe
Intelligent Document Platform, Adobe PDF, Adobe Reader, Adobe Solutions Network, Aldus, Dis-
tiller, ePaper, Extreme, FrameMaker, Illustrator, InDesign, Minion, Myriad, PageMaker, Photo-
shop, Poetica, PostScript, and XMP are either registered trademarks or trademarks of Adobe
Systems Incorporated in the United States and/or other countries. Microsoft and Windows are
either registered trademarks or trademarks of Microsoft Corporation in the United States and/or
other countries. Apple, Mac, Macintosh, and Power Macintosh are trademarks of Apple Computer,
Inc., registered in the United States and other countries. IBM is a registered trademark of IBM
Corporation in the United States. Sun is a trademark or registered trademark of Sun Microsys-
tems, Inc. in the United States and other countries. UNIX is a registered trademark of The Open
Group. SVG is a trademark of the World Wide Web Consortium; marks of the W3C are registered
and held by its host institutions MIT, INRIA and Keio. Helvetica and Times are registered trade-
marks of Linotype-Hell AG and/or its subsidiaries. Arial and Times New Roman are trademarks of
The Monotype Corporation registered in the U.S. Patent and Trademark Office and may be regis-
tered in certain other jurisdictions. ITC Zapf Dingbats is a registered trademark of International
Typeface Corporation. Ryumin Light is a trademark of Morisawa & Co., Ltd. All other trademarks
are the property of their respective owners.
```

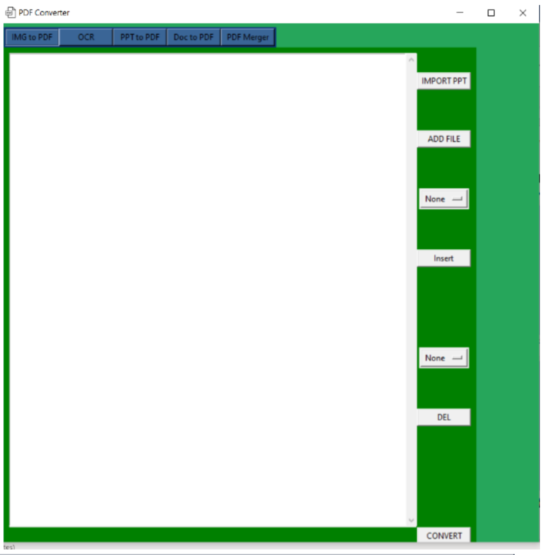
```
Adobe, the Adobe logo, Acrobat, the Acrobat logo, Acrobat Capture, Adobe Garamond, Adobe
Intelligent Docunient Platform, Adobe PDF, Adobe Reader, Adobe Solutions Network, Aldus, Dis-
tiller, ePaper, Extreme, FrameMaker, Illustrator, InDesign, Minion, Myriad, PageMaker, Photo-
shop, Poetica, PostScript, and XMP are either registered trademarks or trademarks of Adobe
Systems Incorporated in the United States and/or other countries. Microsoft and Windows are
either registered trademarks or trademarks of Microsoft Corporation in the United States and/or
other countries. Apple, Mac, Macintosh, and Power Macintosh are trademarks of Apple Computer,
Inc., registered in the United States and other countries. IBM is a registered trademark of IBM
Corporation in the United States. Sun is a trademark or registered trademark of Sun Microsys-
tems, Inc. in the United States and other countries. UNIX is a registered trademark of The Open
Group. SVG is a trademark of the World Wide Web Consortium; marks of the W3C are registered
and held by its host institutions MIT, INRIA and Keio. Helvetica and Times are registered trade-
marks of Linotype-Hell AG and/or its subsidiaries. Arial and Times New Roman are trademarks of
The Monotype Corporation registered in the U.S. Patent and Trademark Office and may be regis-
tered in certain other jurisdictions. ITC Zapf Dingbats is a registered trademark of International
Typeface Corporation. Ryumin Light is a trademark of Morisawa & Co., Ltd. All other trademarks
are the property of their respective owners.
```

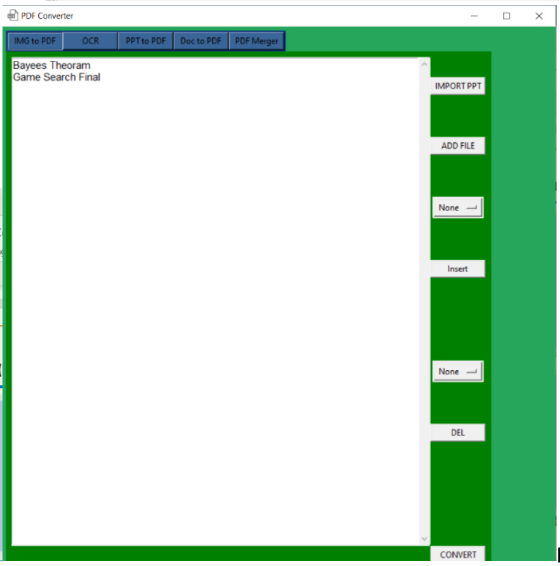
OCR Processed output



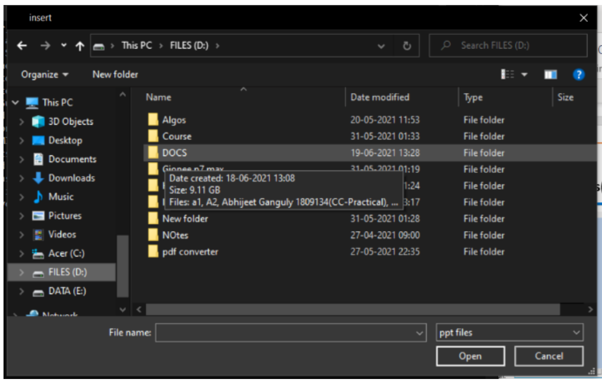
Adobe, the Adobe logo, Acrobat, the Acrobat logo, Acrobat Capture, Adobe Garamond, Adob
Intelligent Document Platform, Adobe PDF, Adobe Reader, Adobe Solutions Network, Aldus, Dis
tiller, ePaper, Extreme, FrameMaker, Illustrator, InDesign, Minion, Myriad, PageMaker, Photo
shop, Poetica, PostScript, and XMP are either registered trademarks or trademarks of Adob
Systems Incorporated in the United States and/or other countries. Microsoft and Windows ar
either registered trademarks or trademarks of Microsoft Corporation in the United States and/o
other countries. Apple, Mac, Macintosh, and Power Macintosh are trademarks of Apple Compute
Inc., registered in the United States and other countries. IBM is a registered trademark of IBN
Corporation in the United States. Sun is a trademark or registered trademark of Sun Microsys
tems, Inc. in the United States and other countries. UNIX is a registered trademark of The Ope
Group. SVG is a trademark of the World Wide Web Consortium; marks of the W3C are registere
and held by its host institutions MIT, INRIA and Keio. Helvetica and Times are registered trade
marks of Linotype-Hell AG and/or its subsidiaries. Arial and Times New Roman are trademarks c
The Monotype Corporation registered in the U.S. Patent and Trademark Office and may be regis
tered in certain other jurisdictions. ITC Zapf Dingbats is a registered trademark of Internationa
Typeface Corporation. Ryumin Light is a trademark of Morisawa & Co., Ltd. All other trademark
are the property of their respective owners.

maxoAdobe, the Adobe logo, Acrobat, the Acrobat logo, Acrobat Capture, Adobe Garamon
d, Adobe
Intelligent Document Platform, Adobe PDF, Adobe Reader, Adobe Solutions Network, Aldu
s, Dis-
tiller, ePaper, Extreme, FrameMaker, Illustrator, InDesign, Minion, Myriad, PageMaker, Phot
o-
shop, Poetica, PostScript, and XMP are either registered trademarks or trademarks of Ado
be
Systems Incorporated in the United States and/or other countries. Microsoft and Windows
are
either registered trademarks or trademarks of Microsoft Corporation in the United States a
nd/or
other countries. Apple, Mac, Macintosh, and Power Macintosh are trademarks of Apple Co
mputer,
Inc., registered in the United States and other countries. IBM is a registered trademark of I
BM.
Corporation in the United States. Sun is a trademark or registered trademark of Sun Micro
sys-
tems, Inc. in the United States and other countries. UNIX is a registered trademark of The
Open





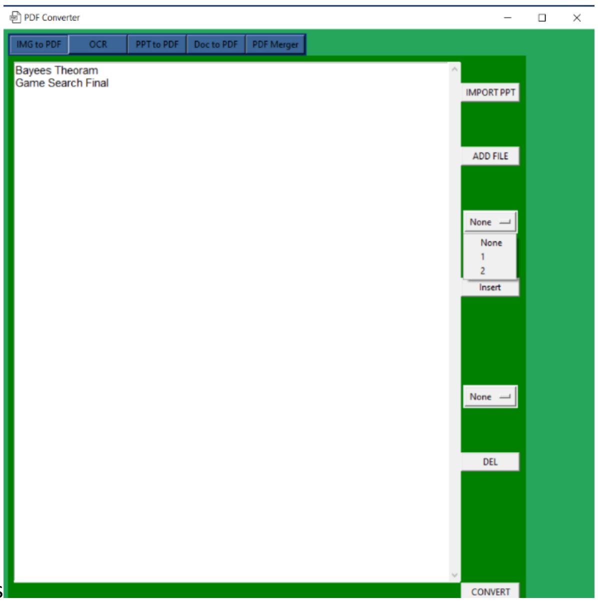
Bayees Theoram
Game Search Final



Name	Date modified	Type	Size
Algos	20-05-2021 11:53	File folder	
Course	31-05-2021 01:33	File folder	
DOCS	19-06-2021 13:28	File folder	
Gionee n7 max	31-05-2021 01:19	File folder	
[illegible]	1:24	File folder	
[illegible]	3:17	File folder	
New folder	31-05-2021 01:28	File folder	
NOtes	27-04-2021 09:00	File folder	
pdf converter	27-05-2021 22:35	File folder	

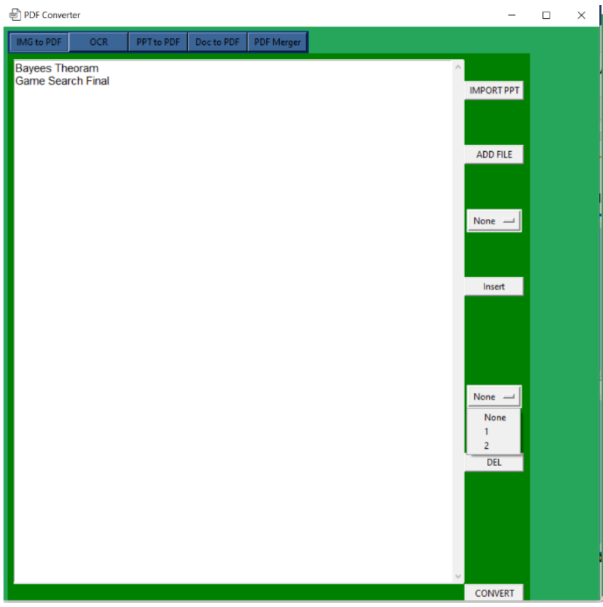
Date created: 18-06-2021 13:08
Size: 9.11 GB
Files: a1, A2, Abhijeet Ganguly 1809134(CC-Practical), ...

PPT

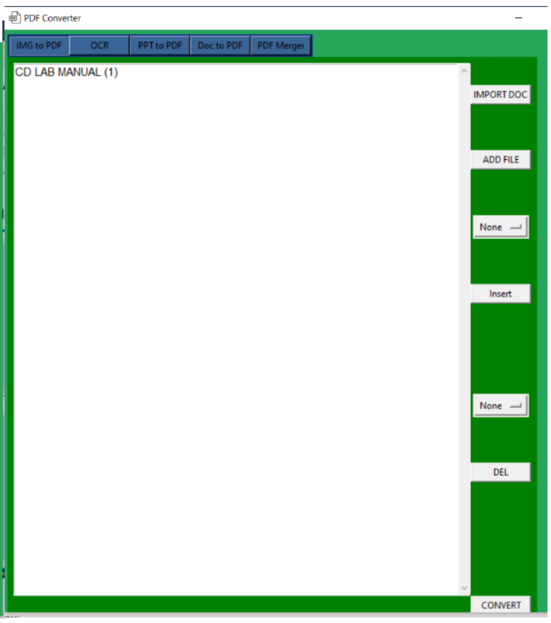


Bayees Theoram
Game Search Final

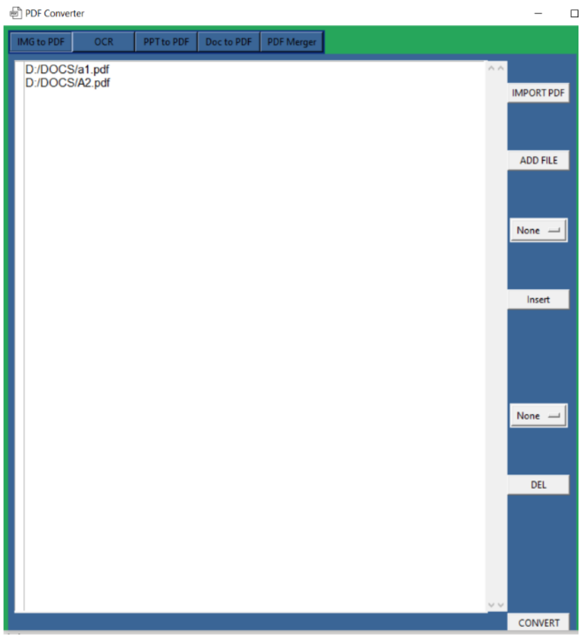
view and imported files



Bayees Theoram
Game Search Final

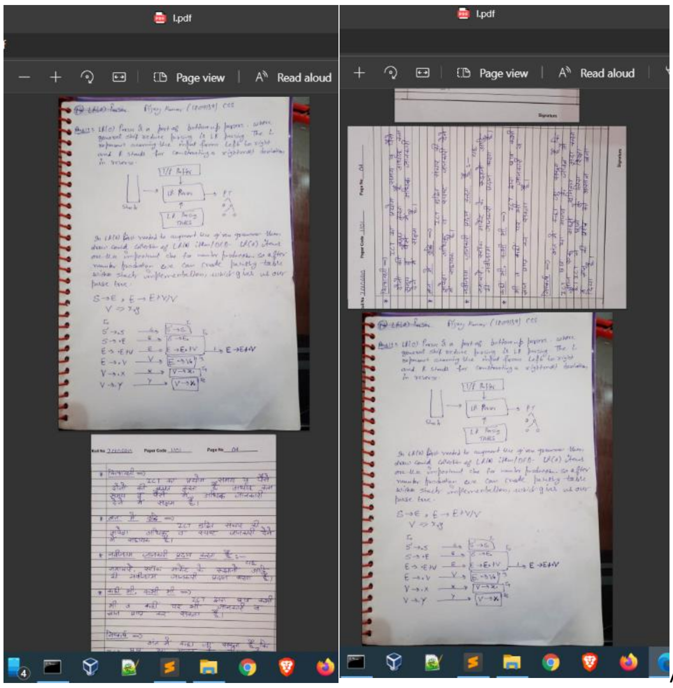


CD LAB MANUAL (1)



D:/DOCS/a1.pdf
D:/DOCS/A2.pdf

Pdf merger main view



After saving pdf in pdf viewer

Features Provided:

This application is oriented to provide many facility related to pdf and images conversion. IMG to PDF , PDFs merging, Image OCR are the visible function of current project.

Current Status:

Currently Project is fully oriented to IMG to PDF where different functionality are :

Crop images

Delete image

Color effects

Rotate image

Save as a Pdf

Merge Multiple PDF files in single pdf file

Converts PPT in to PDF

Converts WORD in to PDF

OCR extracts

Future Goals/Updates :

Updates:

Our application is still under development. Below listed are the Goals for application.

Color Filters

PDF viewer

PDF Compresser

Future Goals:

From our Observation and Research we come to conclusion that Online Platform will be more efficient and easy for users. So we are moving toward a Website and a mobile application development so that everyone can easily use our services.

Mobile application development

Web App development

Applications of Project:

1) **Feature oriented**
   a) This application can be used to make pdfs from raw images.
   b) To crop images
   c) To convert images in pdf
   d) To combine multiple images in single pdf
   e) To merge Pdfs
   f) To perform OCR

2) **Target Users**
   a) Students who need to manage various documents. Also to submit online test sheets (in present study culture)

b) Teachers who need to publish various documents.
c) Any one who need to perform image to pdf operations
d) Developers for well documentation.

Learning outcomes:

- A better understanding of python and its components
- Understanding of terminal and libraries.
- Familiarity with opencv
- Better visualization of system structure.
- Got experience in numpy , cv , pillow, sys ,etc
- Deep understanding of development procedures.