Architecture of Image\_To\_Text\_Convertor

* **from tkinter import filedialog**

Creating the File Explorer

In order to do so, we have to import the filedialog module from Tkinter. The File dialog module will help you open, save files or directories.  
In order to open a file explorer, we have to use the method, askopenfilename(). This function creates a file dialog object.

* **import pytesseract**

Python-tesseract is a wrapper for [Google’s Tesseract-OCR Engine](https://github.com/tesseract-ocr/tesseract). It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

* **import cv2**

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.  
cv2.imread() method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

* **import tkinter as tk**

Tk is the original GUI library for the Tcl language. Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. There are several other popular Python GUI toolkits. Most popular are wxPython, PyQt, and PyGTK.

# specify tesseract.exe file location

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

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images.

Python-tesseract is a wrapper for [Google’s Tesseract-OCR Engine](https://github.com/tesseract-ocr/tesseract). It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

window = tk.Tk ()

window.geometry('455x600')

window.resizable(False, False)

window.title('IMAGE TO TEXT CONVERTER')

window.configure(bg='#B0C4DE')

get\_text = tk.StringVar()

# select\_image function to select image from directory

def select\_image():

"""Making image variable global"""

global image

"""filedialog is used to select the image"""

fileName = filedialog.askopenfilename(initialdir="/", title="Select a Image",

filetype=(

('All Files', '\*.\*'), ("jpeg", "\*.jpeg"), ("png", "\*.png"),

('jpg', '\*.jpg'),))

"""opencv library is used to read image from selected directory"""

image = cv2.imread(fileName)

scale\_percent = 50 # percent of original size

width = int(image.shape[1] \* scale\_percent / 100)

height = int(image.shape[0] \* scale\_percent / 100)

dimensions = (width, height)

# resize image

image\_resized = cv2.resize(image, dimensions, interpolation=cv2.INTER\_AREA)

cv2.imshow('resized', image\_resized)

# This function is used to extract from image

def extract\_text():

global extract\_data

extract\_data = pytesseract.image\_to\_string(image)

value = extract\_data

result = retrive\_text.insert(tk.END, value)

# Text Widget used to store text from image

retrive\_text = tk.Text(window, height=15, width=40, bg='#f5f5f5', font='Arial')

retrive\_text.grid(row=0, column=0, padx=5, pady=5)

# This function is used to save the extracted text from image and save in file in different format

def save\_text():

select\_file\_type = filedialog.asksaveasfilename(initialdir='/',

filetypes=[('text', '.txt'), ('pdf', '.pdf'), ('word', '.doc'),

('excel', '.xls'), ('All Files', '\*.\*')],

defaultextension=".txt")

with open(select\_file\_type, "w") as file:

file.write(extract\_data)

file.close()

# Button Widget in Tkinter

Show\_image\_button = tk.Button(window, text='Select Image', command=select\_image, font=5, bg='#dcdcdc').grid(row=1, column=0, padx=10, pady=10)

# Button Widget in Tkinter

Extract\_text\_button = tk.Button(window, text='Extract Text', command=extract\_text, font=5, bg='#dcdcdc').grid(row=2, column=0, padx=10, pady=10)

# Button Widget in Tkinter

Save\_text\_button = tk.Button(window, text='Save Text', command=save\_text, font=5, bg='#dcdcdc').grid(row=3, column=0, padx=10, pady=10)

# Button Widget in Tkinter

Exit\_button = tk.Button(window, text='Exit', command=exit, font=5, bg='#dcdcdc').grid(row=4, column=0, padx=10, pady=10)

# main window screen will stay visible in screen until its closed by user

window.mainloop()