

## Table Of Contents

S.No	Content	Page No
1	Abstract	2
2	Introduction	3-5
3	Methodology	6-9
4	System Requirements	10-11
5	System Implementation	12-15
6	Code Execution and Screenshots	16-32
7	Testing and Validation	33-36
8	Conclusion	37-38
9	Future Enhancements	39-40
10	Bibliography	41-42

### Abstract

- “Text extraction from image using python” is a python based project which makes uses of OCR concept. OCR (Optical Character Recognition) is the process of electronic conversion of Digital images into machine-encoded text. The digital image is generally an image that contains regions that resemble characters of a language. OCR is a field of research in pattern recognition, artificial intelligence and computer vision. This technique of extracting text from images is generally carried out in work environments where it is certain that the image would be containing text data. For enabling our python program to have Character recognition capabilities, we would be making use of *pytesseract* OCR library. In this project we are making use of *tkinter* which is a GUI library in python. We are providing the way to select an image file in types of jpg and jpeg and png extracting text from those images and finally storing it in text file. This process is done by user involvement using GUI interface.

# **Chapter-1**

## **Introduction**

# 1. Introduction

## 1.1 Optical Character Recognition

OCR = Optical Character Recognition. In other words, OCR systems transform a two-dimensional image of text, that could contain machine printed or handwritten text from its image representation into machine-readable text. OCR as a process generally consists of several sub-processes to perform as accurately as possible. The sub processes are:

- Preprocessing of the Image
- Text Localization
- Character Segmentation
- Character Recognition
- Post Processing

## 1.2 Pytesseract – OCR

Tesseract is open source text recognition (OCR) Engine, available under the Apache 2.0 license. It can be used directly, or (for programmers) using an API to extract printed text from images. It supports a wide variety of languages. Tesseract doesn't have a built-in GUI. Tesseract is compatible with many programming languages and frameworks through wrappers that can be found [here](#). It can be used with the existing layout analysis to recognize text within a large document, or it can be used in conjunction with an external text detector to recognize text from an image of a single text line.

### 1.3 Advantages

The following are the advantages of this project:

- This project helps us to extract text from the images such as YouTube tutorial snippets, textbook images and so on.
- This project saves time by extracting text directly from images without typing.
- This project also helps to overcome the typing mistakes as we extract text directly from images.

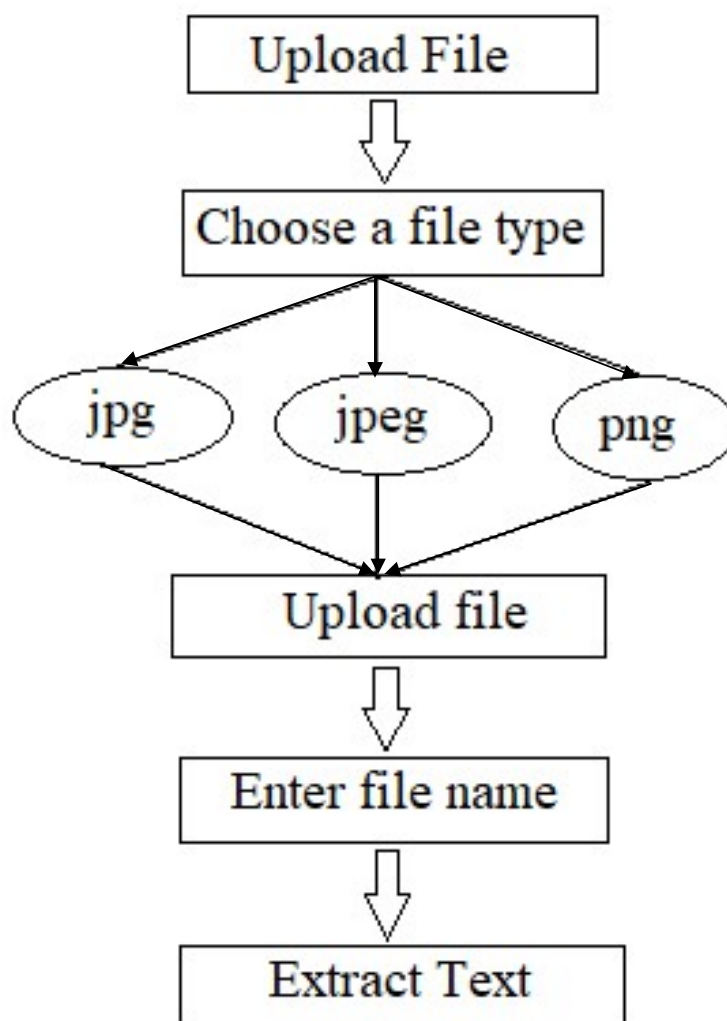
# **Chapter-2**

## **Methodology**

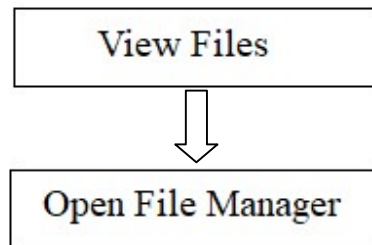
## 2. Methodology

### 2.1 Flowchart

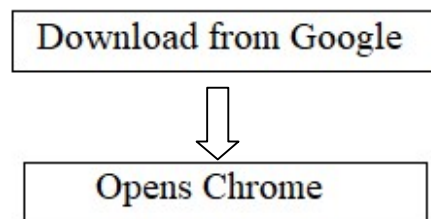
#### 2.1.1 Upload File



### 2.1.2 View Files



### 2.1.2 Download from Google



## 2.2 Modules

**Pytesseract** - Pytesseract or Python-tesseract is an OCR tool for python that also serves as a wrapper for the Tesseract-OCR Engine. It can read and recognize text in images and is commonly used in python ocr image to text use cases.

**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.

**PIL** - PIL is the Python Imaging Library which provides the python interpreter with image editing capabilities. The Image module provides a class with the same name which is used to represent a PIL image.



## Text Extraction from Image using Python

---

**OS** - The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc. You first need to import the os module to interact with the underlying operating system.

# **Chapter-3**

## **System Requirements**

## **3. System Requirements**

### **3.1 Hardware Requirements**

- Processor - 5th Generation Intel(i3, i5,i7)
- RAM - 4 GB(min)
- Hard Disk - 1 TB
- Keyboard - Standard Windows Keyboard
- Mouse - Two or Three Button
- Display - 1280 X 800 (minimum resolution)

### **3.2 Software requirements**

- Operating System - Microsoft Windows
- Application Software - Pycharm Software
- Language Used – Python

# **Chapter-4**

## **System Implementation**

## 4. System Implementation

### Python

Python is a high-level, general-purpose and a very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry. Python Programming Language is very well suited for Beginners, also for experienced programmers with other programming languages like C++ and Java.

This specially designed Python tutorial will help you learn Python Programming Language in most efficient way, with the topics from basics to advanced (like Web- scraping, Django, Deep-Learning, etc.) with examples.

Below are some facts about Python Programming Language:

1. Python is currently the most widely used multi-purpose, high-level programming language.
2. Python allows programming in Object-Oriented and Procedural paradigms.
3. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
4. Python language is being used by almost all tech-giant companies like – Google, Amazon, Face book, Instagram, Drop box, Uber... etc.
5. The biggest strength of Python is huge collection of standard library which can be used for the following:
  - Machine Learning
  - GUI Applications (like Tkinter, PyQt etc. )
  - Web frameworks like Django (used by YouTube, Instagram, Dropbox)

## Text Extraction from Image using Python

---

- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia
- Scientific computing
- Text processing and many more.

**Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

**Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

**Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

**Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

### History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

**Python is copyrighted** - Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

### Python Features

Python's features include –

## Text Extraction from Image using Python

---

**Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

**Easy-to-read** – Python code is more clearly defined and visible to the eyes.

**Easy-to-maintain** – Python's source code is fairly easy-to-maintain.

**A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

**Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

**Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

**Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

**Databases** – Python provides interfaces to all major commercial databases.

**GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below -

1. It supports functional and structured programming methods as well as OOP.
2. It can be used as a scripting language or can be compiled to byte-code for building large applications.
3. It provides very high-level dynamic data types and supports dynamic type checking.
4. It supports automatic garbage collection.
5. It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

# **Chapter-5**

## **Code Execution and Screenshots**



### Code:

#### Methods:

- upload()
- view()
- download()
- \_init\_()
- select\_file()
- upload\_file()

#### Classes:

- Text\_Extraction

### main.py:

```
import os

from subprocess import call

from tkinter import ttk, filedialog

import webbrowser

import sys

try:

    from Tkinter import *

except ImportError:

    from tkinter import *

try:
```

## Text Extraction from Image using Python

---

```
import ttk

py3 = False

except ImportError:

    import tkinter.ttk as ttk

    py3 = True

def upload():

    call(["python","upload.py"])

def view():

    f_types = [('Text Files','*.txt')]

    filename = filedialog.askopenfilename(filetypes=f_types)

    os.startfile(filename)

def download():

    webbrowser.open_new_tab("http://www.google.com")

class Text_Extraction:

    def __init__(self):

        root = Tk()

        """This class configures and populates the toplevel window.

        top is the toplevel containing window."""

        _bgcolor = '#d9d9d9' # X11 color: 'gray85'

        _fgcolor = '#000000' # X11 color: 'black'

        _compcolor = '#ffffff' # X11 color: 'white'
```

## Text Extraction from Image using Python

---

```
_ana1color = '#ffffff' # X11 color: 'white'

_ana2color = '#ffffff' # X11 color: 'white'

font14 = "-family {Segoe UI} -size 15 -weight bold -slant " \
"roman -underline 0 -overstrike 0"

font16 = "-family {Swis721 BlkCn BT} -size 40 -weight bold " \
"-slant roman -underline 0 -overstrike 0"

font9 = "-family {Segoe UI} -size 9 -weight normal -slant " \
"roman -underline 0 -overstrike 0"

root.geometry("963x600")

root.title("Text Extraction")

root.configure(background="#d9d9d9")

root.configure(highlightbackground="#d9d9d9")

root.configure(highlightcolor="black")

self.menubar = Menu(root,font=font9,bg=_bgcolor,fg=_fgcolor)

root.configure(menu = self.menubar)

self.Frame1 = Frame(root)

self.Frame1.place(relx=0.02, rely=0.03, relheight=0.94, relwidth=0.96)

self.Frame1.configure(relief=GROOVE)

self.Frame1.configure(borderwidth="10")

self.Frame1.configure(relief=GROOVE)

self.Frame1.configure(background="#d9d9d9")
```

## Text Extraction from Image using Python

---

```
self.Frame1.configure(highlightbackground="#d9d9d9")

self.Frame1.configure(highlightcolor="black")

self.Frame1.configure(width=925)

self.Message1 = Message(self.Frame1)

self.Message1.place(relx=0.09, rely=0.01, relheight=0.15, relwidth=0.86)

self.Message1.configure(background="#d9d9d9")

self.Message1.configure(font=font16)

self.Message1.configure(foreground="#000000")

self.Message1.configure(highlightbackground="#d9d9d9")

self.Message1.configure(highlightcolor="black")

self.Message1.configure(text="Text Extraction From Image")

self.Message1.configure(width=791)

self.Button2 = Button(self.Frame1)

self.Button2.place(relx=0.18, rely=0.17, height=103, width=566)

self.Button2.configure(activebackground="#d9d9d9")

self.Button2.configure(activeforeground="#000000")

self.Button2.configure(background="#d9d9d9")

self.Button2.configure(disabledforeground="#bfbfbf")

self.Button2.configure(font=font14)

self.Button2.configure(foreground="#000000")

self.Button2.configure(highlightbackground="#d9d9d9")
```

## Text Extraction from Image using Python

---

```
self.Button2.configure(highlightcolor="black")

self.Button2.configure(pady="0")

self.Button2.configure(text="Upload File")

self.Button2.configure(width=566)

self.Button2.configure(command=upload)

self.Button3 = Button(self.Frame1)

self.Button3.place(relx=0.18, rely=0.40, height=93, width=566)

self.Button3.configure(activebackground="#d9d9d9")

self.Button3.configure(activeforeground="#000000")

self.Button3.configure(background="#d9d9d9")

self.Button3.configure(disabledforeground="#bfbfbf")

self.Button3.configure(font=font14)

self.Button3.configure(foreground="#000000")

self.Button3.configure(highlightbackground="#d9d9d9")

self.Button3.configure(highlightcolor="black")

self.Button3.configure(pady="0")

self.Button3.configure(text="View Files")

self.Button3.configure(width=566)

self.Button3.configure(command=view)

self.Button4 = Button(self.Frame1)

self.Button4.place(relx=0.18, rely=0.63, height=103, width=566)
```

## Text Extraction from Image using Python

---

```
self.Button4.configure(activebackground="#d9d9d9")

self.Button4.configure(activeforeground="#000000")

self.Button4.configure(background="#d9d9d9")

self.Button4.configure(disabledforeground="#bfbfbf")

self.Button4.configure(font=font14)

self.Button4.configure(foreground="#000000")

self.Button4.configure(highlightbackground="#d9d9d9")

self.Button4.configure(highlightcolor="black")

self.Button4.configure(pady="0")

self.Button4.configure(text="Download from Google")

self.Button4.configure(width=566)

self.Button4.configure(command=download)

root.mainloop()

if __name__ == '__main__':

    GUUEST=Text_Extraction()
```

### **upload.py:**

```
from pytesseract import pytesseract

from tkinter import *

from tkinter import ttk, filedialog

from PIL import ImageTk, Image
```

## Text Extraction from Image using Python

---

```
import tkinter as tk

import subprocess

def select_file():

    global entry,img

    b='false'

    string= entry.get()

    if(string==""):

        l4 = Label(bottomframe, text = "Enter a file name")

        l4.config(font=("roman -underline 0 -overstrike 0", 10),foreground="red")

        l4.pack()

    else:

        path_to_tesseract = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

        pytesseract.tesseract_cmd = path_to_tesseract

        try:

            text = pytesseract.image_to_string(img)

            b='true'

        except:

            l4 = Label(bottomframe, text = "Upload a file")

            l4.config(font=("roman -underline 0 -overstrike 0", 10),foreground="red")

            l4.pack()

            #label.configure(text=string)
```

## Text Extraction from Image using Python

---

```
if(b=='true'):

    name=string+".txt"

    file = open(name, "a")

    # Appending the text into file

    file.write(text)

    file.write("\n")

    # Close the file

    file.close

    file = open(name)

    # Displaying the extracted text

    print(text[:-1])

    l4 = Label(bottomframe, text = "Text is successfully Extracted")

    l4.config(font=("Courier 22 bold", 20),foreground="green")

    l4.pack()

    subprocess.call(['notepad.exe', name])

def upload_file():

    global img

    f_types=['null']

    if(v.get()==1):

        f_types = [('Jpg Files','*.jpg')]

    elif(v.get()==2):
```



## Text Extraction from Image using Python

---

```
f_types = [('Jpeg Files','*.jpeg')]

elif(v.get()==3):

    f_types = [('Png Files','*.png')]

else:

    l3 = Label(bottomframe, text = 'Choose file type')

    l3.config(font=("roman -underline 0 -overstrike 0", 10),foreground="red")

    l3.pack()


# Passing the image object to image_to_string() function

# This function will extract the text from the image

if(f_types!=['null']):

    filename = filedialog.askopenfilename(filetypes=f_types)

    img = Image.open(filename)

    s=filename.split("/")

    string1=s[-1]

    label.configure(text=string1)

    #l3 = Label(root, text = s[-1])

    #l3.config(font=("roman -underline 0 -overstrike 0", 10))

    #l3.pack()

root = Tk()

root.geometry("963x600")
```

## Text Extraction from Image using Python

---

```
root.title("Text Extraction")

T = Text(root, height = 5, width = 52)

# Create label

l = Label(root, text = "Text Extraction From Image")

l.config(font = ("Courier", 40))

l.pack()

v = tk.IntVar()

l1 = Label(root, text = "Choose a file type:")

l1.config(font = ("roman -underline 0 -overstrike 0", 20))

l1.pack()

tk.Radiobutton(root,

                text="Jpg",

                font = ("roman -underline 0 -overstrike 0", 20),

                padx = 20,

                variable=v,

                value=1).pack(anchor=tk.CENTER)

tk.Radiobutton(root,

                text="Jpeg",

                font = ("roman -underline 0 -overstrike 0", 20),

                padx = 20,

                variable=v,
```

## Text Extraction from Image using Python

---

```
value=2).pack(anchor=tk.CENTER)

tk.Radiobutton(root,

    text="Png",

    font=("roman -underline 0 -overstrike 0", 20),

    padx = 20,

    variable=v,

    value=3).pack(anchor=tk.CENTER)

l2 = Label(root, text = "Upload file : Click the below button")

l2.config(font=("roman -underline 0 -overstrike 0", 20))

l2.pack()

btn = Button(root, text='Click Here', width=8,

    height=1, bd='10',font=("Helvetica", 15),bg="White", command=upload_file)

btn.pack()

label=Label(root, text="", font=("Courier 22 bold"))

label.pack()

#Create an Entry widget to accept User Input

entry= Entry(root, width= 40, bd=5)

entry.focus_set()

entry.pack()

btn1 = Button(root, text='Click Here', width=8,

    height=1, bd='10',font=("Helvetica", 15),bg="White", command=select_file)
```

## Text Extraction from Image using Python

---

```
btn1.pack()

bottomframe = Frame(root)

bottomframe.pack( side = BOTTOM )

root.mainloop()
```

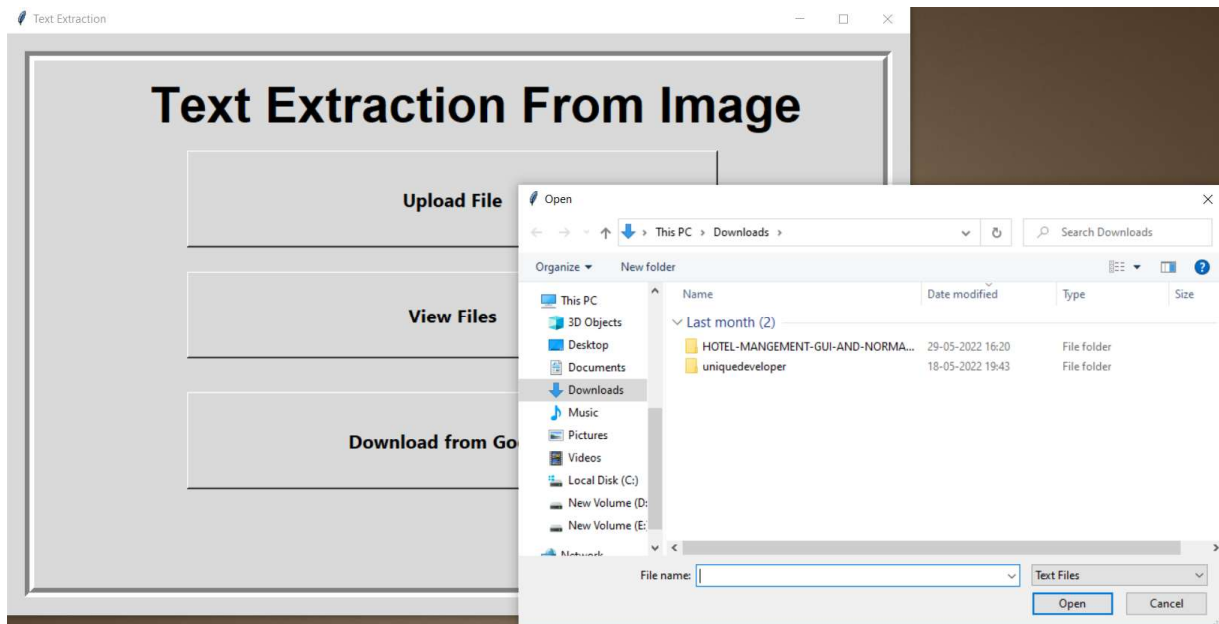
## Screenshots

**main.py**

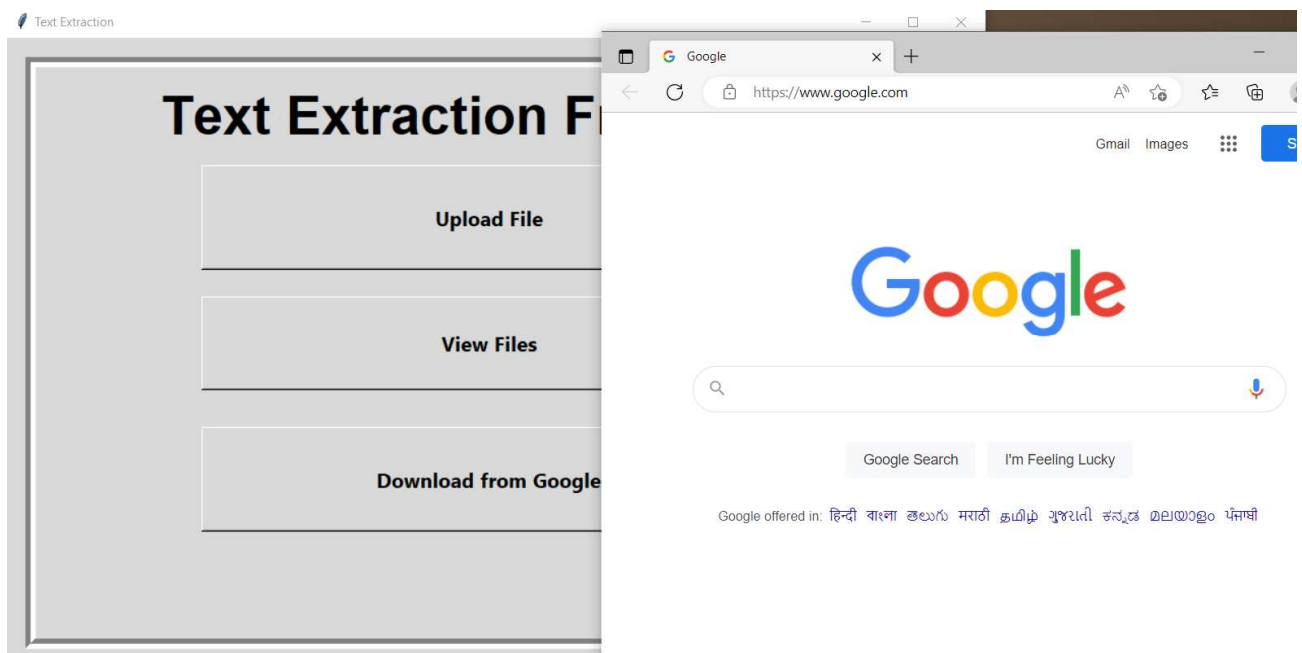


**Fig1: Opening Screen**

## Text Extraction from Image using Python

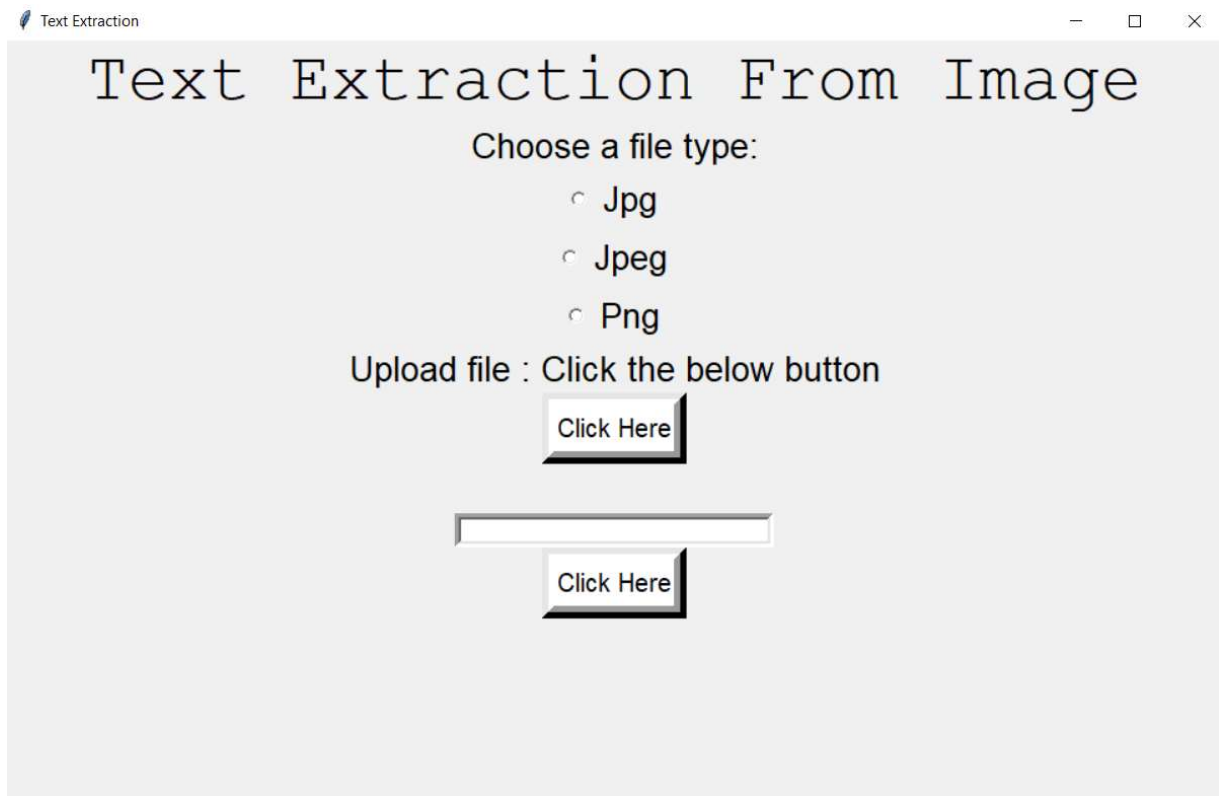


**Fig2: When View files is clicked**



**Fig3: When Download from Google is clicked**

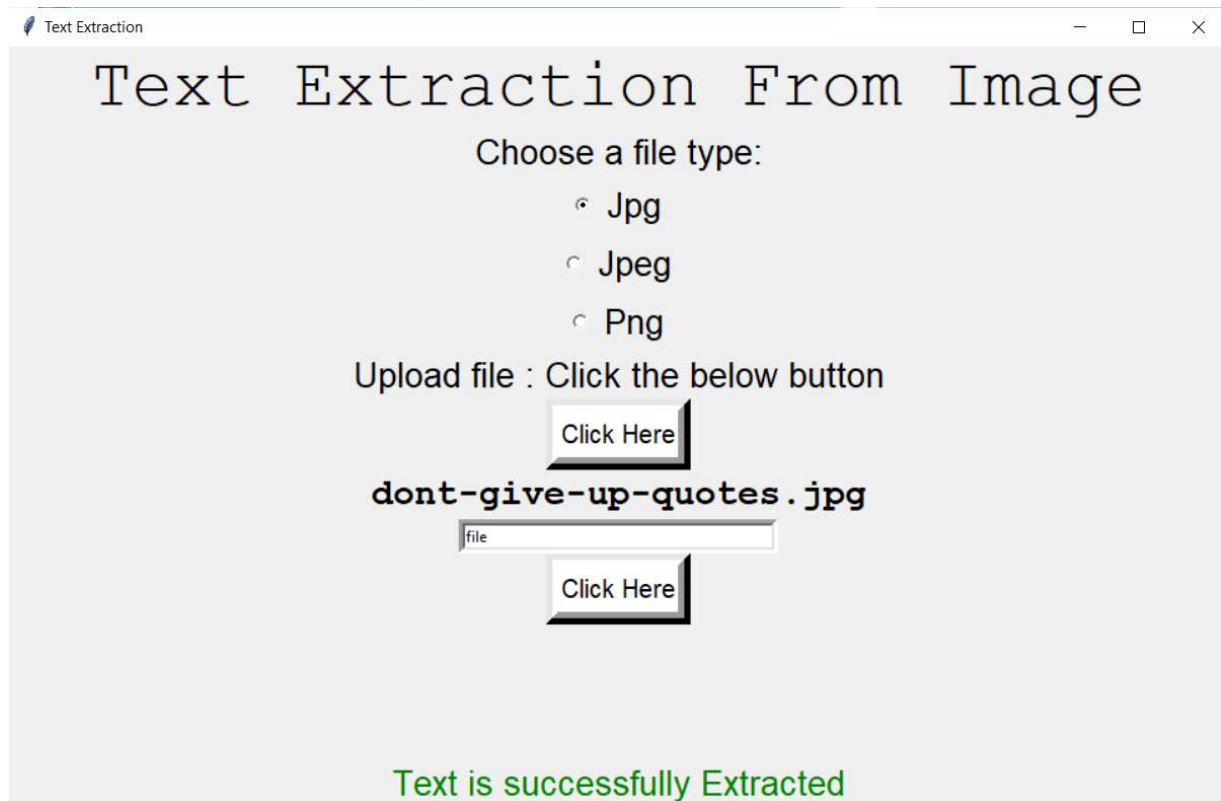
### upload.py



**Fig4: When Upload file is clicked**

## Text Extraction from Image using Python

---



Text Extraction From Image

Choose a file type:

☒ Jpg  
☐ Jpeg  
☐ Png

Upload file : Click the below button

Click Here

**dont-give-up-quotes.jpg**

file

Click Here

**Text is successfully Extracted**

**Fig5: When the text is extracted**



# **Chapter-6**

## **Testing and Validation**

### 6. Testing and Validation

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

#### **Testing Objectives include:**

Testing is a process of executing a program with the intent of finding an error.

A good test case is one that has a probability of finding an as yet undiscovered error.

A successful test is one that uncovers an undiscovered error.

#### **Testing Principles:**

All tests should be traceable to end user requirements

- Tests should be planned long before testing begins
- Testing should begin on a small scale and progress towards testing in large
- Exhaustive testing is not possible
- To be most effective testing should be conducted by an independent third party

A Strategy for software testing integrates software test cases into a series of well-planned steps that result in the successful construction of software. Software testing is a broader topic for what is referred to as Verification and Validation. Verification refers to the set of activities that ensure that the software correctly implements a specific function. Validation refers to the set of activities that ensure that the software that has been built is traceable to customer's requirements.

#### **Types of Testing:**

##### **Unit tests**

Unit tests are very low level and close to the source of an application. They consist in testing individual methods and functions of the classes, components, or modules used by your

## Text Extraction from Image using Python

---

software. Unit tests are generally quite cheap to automate and can run very quickly by a continuous integration server.

### **Integration tests**

Integration tests verify that different modules or services used by your application work well together. For example, it can be testing the interaction with the database or making sure that microservices work together as expected. These types of tests are more expensive to run as they require multiple parts of the application to be up and running.

### **Functional tests**

Functional tests focus on the business requirements of an application. They only verify the output of an action and do not check the intermediate states of the system when performing that action.

There is sometimes confusion between integration tests and functional tests as they both require multiple components to interact with each other. The difference is that an integration test may simply verify that you can query the database while a functional test would expect to get a specific value from the database as defined by the product requirements.

### **System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### **White Box Testing**

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

### **Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests,

## Text Extraction from Image using Python

---

must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

### **Acceptance testing**

Acceptance tests are formal tests that verify if a system satisfies business requirements. They require the entire application to be running while testing and focus on replicating user behaviors. But they can also go further and measure the performance of the system and reject changes if certain goals are not met.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

# **Chapter-7**

# **Conclusion**

### 7.Conclusion

This project, “Text Extraction from Image using Python”, is very useful for those who want to make use of the text embedded in an image. This project helps us to get text from the screen snippets such as YouTube and from the images of a book and so on. This project is very useful as it saves a lot of time wasted during typing and also it reduces typing errors. Thus this project is very useful for students, teachers and also for those who want to extract text from images.

So we have concluded that:

- The software is working properly
- All the features are working as per the requirements
- The text extraction is successfully done
- The software has passed all types of testing
- The software is accepting jpg, jpeg, png image formats
- The text is automatically getting stored in the specified file

# **Chapter-8**

## **Future Enhancements**

## 8. Future Enhancements

- We can make this project as a web application, so we can use this project without installing.
- Thus we can use the project from anywhere and from any system.
- We can even enhance this project to identify the texts other than English language.
- We can add more beautiful user-interfaces, thus making this project visually attractive.

Finally, we can make this project more flexible, user-friendly and more attractive.



# **Chapter-9**

## **Bibliography**

## 9. Bibliography

- [1] <https://www.etutorialspoint.com/index.php/381-text-extraction-from-image-using-opencv-and-ocr-python>
- [2] <https://stackhowto.com/how-to-open-a-new-window-with-a-button-in-python-tkinter/>
- [3] <https://stackoverflow.com/questions/7775001/add-radio-button-to-tkinter>
- [4] <https://www.c-sharpcorner.com/article/how-to-add-frame-in-tkinter-in-python/>
- [5] <https://www.geeksforgeeks.org/how-to-add-padding-to-a-tkinter-widget-only-on-one-side/>
- [6] <https://python-course.eu/tkinter/radio-buttons-in-tkinter.php>
- [7] [https://www.geeksforgeeks.org/setting-the-position-of-tkinter-labels/#:~:text=We%20can%20use%20place\(\),position%20of%20the%20Tkinter%20labels.](https://www.geeksforgeeks.org/setting-the-position-of-tkinter-labels/#:~:text=We%20can%20use%20place(),position%20of%20the%20Tkinter%20labels.)
- [8] <https://www.geeksforgeeks.org/change-the-color-of-certain-words-in-the-tkinter-text-widget/>
- [9] <https://stackoverflow.com/questions/22445217/python-webbrowser-open-to-open-chrome-browser>
- [10] <https://www.geeksforgeeks.org/how-to-get-the-input-from-tkinter-text-box/>
- [11] [https://www.w3schools.com/python/python\\_try\\_except.asp](https://www.w3schools.com/python/python_try_except.asp)

