# Image Viewer Application

## Abstract

This document describes the implementation of a simple Image Viewer application using Python's tkinter library. The application allows users to select an image file from their computer and display it within the application window. It supports JPG and PNG file formats, with the ability to resize the image to fit the display area. This tool demonstrates the integration of tkinter and PIL for GUI development and image manipulation.

## Introduction

The Image Viewer application is designed to provide a user-friendly interface for viewing images. It uses tkinter, a standard GUI library in Python, to create the graphical interface, and the Pillow library for image processing. This project is aimed at beginners in Python GUI programming and serves as a foundation for more advanced projects. The application allows users to browse and select images from their file system and displays the selected image in a resized format to fit the application window. This document provides the implementation code, explains its features, and outlines its functionality.

## Code Implementation

from tkinter import \*  
from tkinter import filedialog  
import tkinter as tk  
from PIL import Image, ImageTk  
import os  
  
def showimage():  
 """Open and display the selected image."""  
 # Open file dialog to select an image  
 filename = filedialog.askopenfilename(  
 initialdir=os.getcwd(),  
 title="Select Image File",  
 filetypes=(("JPG File", "\*.jpg"), ("PNG File", "\*.png"), ("All Files", "\*.\*"))  
 )  
 if filename:  
 # Open and display the selected image  
 img = Image.open(filename)  
 img = img.resize((300, 300)) # Resize image to fit the window  
 img = ImageTk.PhotoImage(img)  
  
 lbl.configure(image=img)  
 lbl.image = img # Keep a reference to avoid garbage collection  
  
# Create the main tkinter window  
root = Tk()  
  
# Frame for buttons  
frame = Frame(root)  
frame.pack(side=BOTTOM, padx=15, pady=15)  
  
# Label to display the image  
lbl = Label(root)  
lbl.pack()  
  
# Button to select an image  
btn\_select = Button(frame, text="Select Image", command=showimage)  
btn\_select.pack(side=tk.LEFT)  
  
# Button to exit the application  
btn\_exit = Button(frame, text="Exit", command=root.quit)  
btn\_exit.pack(side=tk.LEFT, padx=12)  
  
# Configure the main window  
root.title("Image Viewer")  
root.geometry("400x450")  
root.resizable(False, False)  
  
# Run the main event loop  
root.mainloop()

# QR Code Generator

## Abstract

Abstract:  
This project focuses on creating a graphical user interface (GUI) application using Python for generating and saving QR codes. The application employs the Tkinter library for the GUI, the qrcode module for QR code generation, and PIL for image handling. This tool demonstrates user-friendly features, including text input, QR code preview, and saving options, catering to both technical and non-technical users.

## Introduction

QR codes have become an integral part of digital communication, enabling quick and efficient sharing of information. This project provides a simple yet effective QR Code Generator, built with Python, aimed at offering ease of use and .functional flexibility. With a clean GUI designed using Tkinter, users can input data, generate corresponding QR codes,   
preview them within the application, and save them as image files. Such a tool finds relevance in various domains, including business, education, and personal use, offering a gateway to understanding Python's capabilities in GUI and   
image processing applications.

## Code Implementation

import tkinter as tk

from tkinter import messagebox, filedialog

import qrcode

from PIL import Image, ImageTk

def generate\_qr():

    """Generate a QR code based on the input text."""

    data = entry.get()

    if not data:

        messagebox.showerror("Error", "Please enter some text or a URL!")

        return

    # Generate QR Code

    qr = qrcode.QRCode(version=1, box\_size=10, border=4)

    qr.add\_data(data)

    qr.make(fit=True)

    img = qr.make\_image(fill\_color="black", back\_color="white")

    img.thumbnail((200, 200))  # Resize for preview in GUI

    # Display QR Code in the preview label

    qr\_img = ImageTk.PhotoImage(img)

    qr\_label.configure(image=qr\_img)

    qr\_label.image = qr\_img  # Keep a reference to avoid garbage collection

def save\_qr():

    """Save the generated QR code to a file."""

    data = entry.get()

    if not data:

        messagebox.showerror("Error", "Please generate a QR Code first!")

        return

    # Open file dialog for saving the QR Code

    file\_path = filedialog.asksaveasfilename(

        defaultextension=".png",

        filetypes=[("PNG files", "\*.png"), ("All files", "\*.\*")],

    )

    if file\_path:

        # Generate and save QR Code

        qr = qrcode.QRCode(version=1, box\_size=10, border=4)

        qr.add\_data(data)

        qr.make(fit=True)

        img = qr.make\_image(fill\_color="black", back\_color="white")

        img.save(file\_path)

        messagebox.showinfo("Success", f"QR Code saved to {file\_path}")

# Create the GUI window

root = tk.Tk()

root.title("QR Code Generator")

root.geometry("400x500")

root.resizable(False, False)

# Title Label

title = tk.Label(root, text="QR Code Generator", font=("Helvetica", 18, "bold"))

title.pack(pady=10)

# Input Field

frame = tk.Frame(root)

frame.pack(pady=10)

entry = tk.Entry(frame, width=35, font=("Helvetica", 14))

entry.pack(side=tk.LEFT, padx=5)

generate\_btn = tk.Button(frame, text="Generate", command=generate\_qr)

generate\_btn.pack(side=tk.RIGHT, padx=5)

# QR Code Display

qr\_label = tk.Label(root, text="Your QR Code will appear here", font=("Helvetica", 12), fg="gray")

qr\_label.pack(pady=20)

# Save Button

save\_btn = tk.Button(root, text="Save QR Code", command=save\_qr, font=("Helvetica", 12))

save\_btn.pack(pady=10)

# Run the GUI application

root.mainloop()