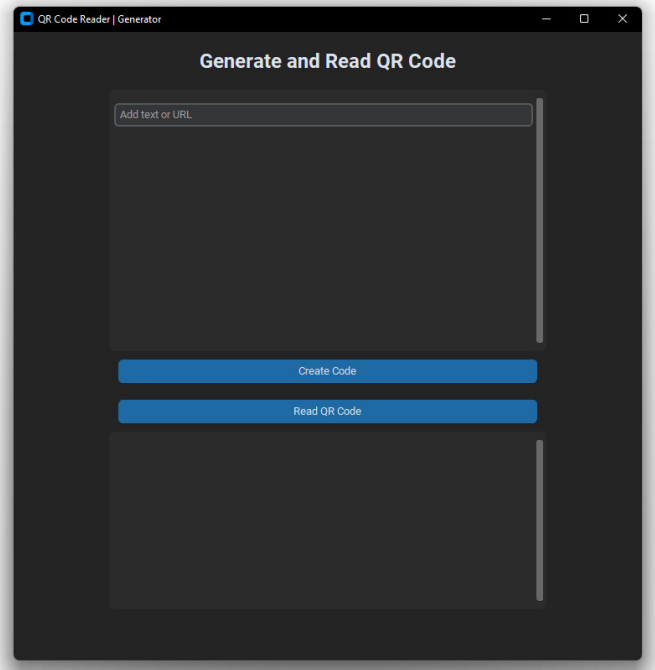
QR CODE
READER | GENERATOR

PYTHON BASED PROJECT
QR Code Reader with Intergrated GUI

Introduction

The QR Code Reader and Generator is a versatile Python program that offers users the ability to create and read QR codes. It makes use of several libraries, including customtkinter and pyqrcode, to create QR codes and display them. The program also employs the pyzbar library to read QR codes from images, making it a powerful tool for users who need to work with QR codes regularly.



One of the key advantages of the QR Code Reader and Generator is its flexibility. With the ability to both create and read QR codes, users can use the program for a wide range of tasks. Additionally, the use of Python and its various libraries allows for easy customization and integration with other tools and workflows. Overall, the QR Code Reader and Generator is a useful program for anyone who works with QR codes on a regular basis.

Architecture

The code is written in Python and uses several libraries, including customtkinter, pyqrcode, png, tkinter, PIL, cv2, numpy, and pyzbar. The customtkinter library is a modified version of the tkinter library that includes additional widgets and features. Pyqrcode is used to generate the QR code image from the user input, while png is used to save the image as a PNG file. Tkinter provides the GUI interface for the code, and PIL is used to display the QR code image in the GUI. CV2 and numpy are used to read the QR code image and decode the data, and pyzbar is used to perform the actual decoding.

The code is designed to perform two main functions: generating and reading QR codes. To generate a QR code, the user inputs text or a URL into an entry box in the GUI, and then clicks a button to create the code. The code is then saved as a PNG file, and the image is displayed in the GUI using PIL. To read a QR code, the user selects an image file containing a QR code, and the code uses CV2 and numpy to read the image and pyzbar to decode the data. The decoded data is then displayed in the GUI.

The GUI is designed using customtkinter widgets and frames. The main window is created using the CTk class from customtkinter, and the title and size of the window are set using the geometry and title methods. The CTkScrollableFrame is used to create a scrollable frame for displaying the decoded data, and the CTkLabel and CTkEntry widgets are used for displaying text and receiving user input. The CTkButton widget is used for the create and read buttons, and the CTkFrame widget is used to contain the QR code image. Overall, the code architecture is well-organized and makes use of a variety of libraries and custom widgets to provide a user-friendly experience for generating and reading QR codes.

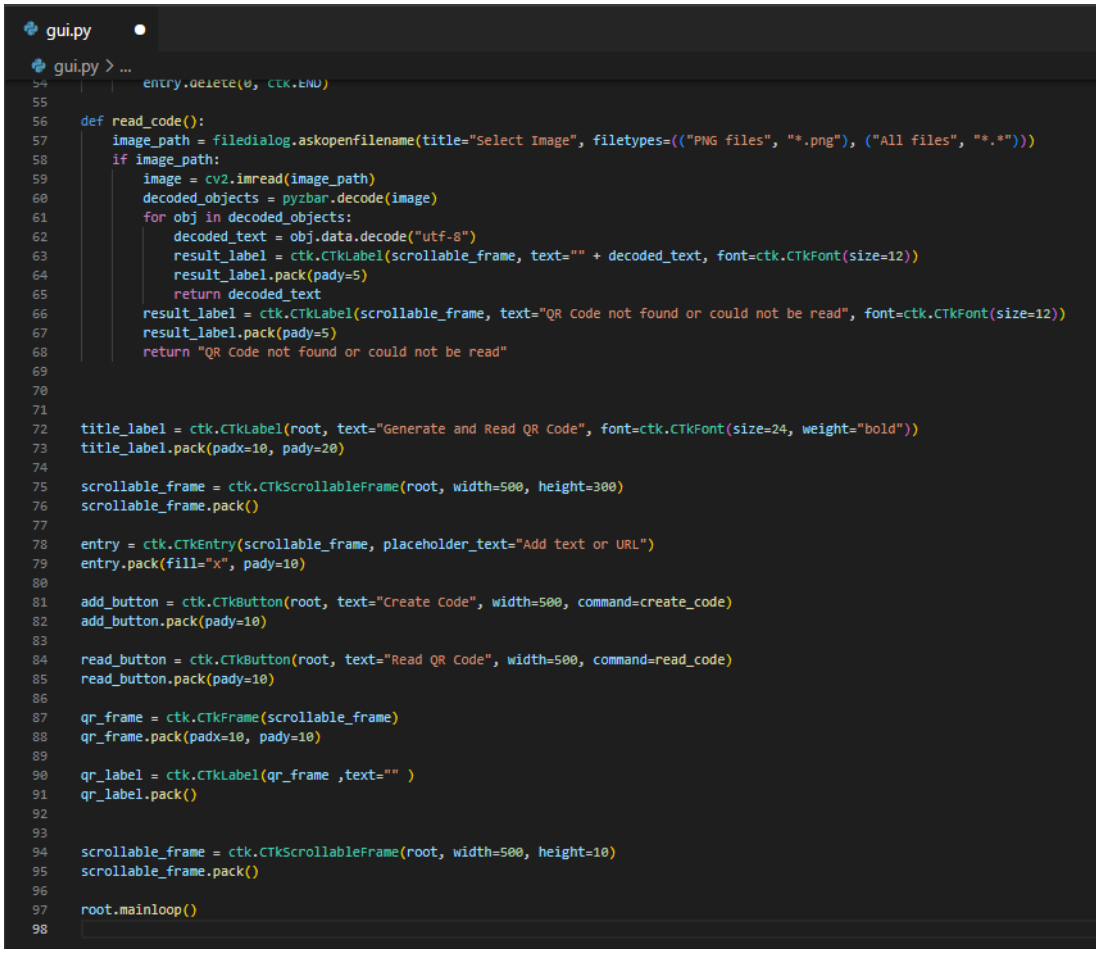
Code Structure

The code is written in Python and has a straightforward structure that is easy to understand. The code starts with importing the necessary libraries like customtkinter, pyqrcode, png, tkinter, PIL, cv2, numpy, and pyzbar. These libraries provide various functionalities, such as creating QR codes, saving them as PNG images, reading QR codes from images, and displaying them on the GUI. The libraries are then used throughout the code to create a functional QR code generator and reader.

The main function of the code is to create and read QR codes. The code structure is divided into two functions, one for creating QR codes and the other for reading them. The create_code() function uses the pyqrcode library to generate QR codes from the text or URL entered by the user in the GUI's entry field. It then saves the QR code as a PNG image using the filedialog library. If the user does not enter a file name with a ".png" extension, the function adds it automatically. Finally, the function displays the QR code image in the GUI. The read_code() function uses the pyzbar library to read QR codes from an image file selected by the user using the filedialog library. If the library can decode the QR code in the image, the function displays the decoded text in a label. If the QR code is not found or cannot be decoded, the function displays an error message.

gui.py

```python
import customtkinter as ctk
import pyqrcode
import png
from tkinter import filedialog
from PIL import Image, ImageTk
import cv2
import numpy as np
import pyzbar.pyzbar as pyzbar


root = ctk.CTk()
root.geometry("750x750")
root.title("QR Code Reader | Generator")

def create_code():
    input_path = filedialog.asksaveasfilename(title="Save Image",
        filetyp=(("PNG File", ".png"), ("All Files","*.*")))

    if input_path:
        if input_path.endswith(".png"):
            get_code =pyqrcode.create(entry.get())
            get_code.png(input_path, scale=8)
        else:
            input_path =f'{input_path}.png'
            get_code =pyqrcode.create(entry.get())
            get_code.png(input_path=8)

        global get_image
        get_image = ImageTk.PhotoImage(Image.open(input_path))
        qr_label.configure(image=get_image, padx=20, pady=20)
        finished_label = ctk.CTkLabel(qr_frame, text="Finished", font=ctk.CTkFont(size=12))
        finished_label.pack(pady=10)
        entry.delete(0, ctk.END)


def create_code():
    input_path = filedialog.asksaveasfilename(title="Save Image",
        filetyp=(("PNG File", ".png"), ("All Files","*.*")))

    if input_path:
        if input_path.endswith(".png"):
            get_code =pyqrcode.create(entry.get())
            get_code.png(input_path, scale=8)
        else:
            input_path =f'{input_path}.png'
            get_code =pyqrcode.create(entry.get())
            get_code.png(input_path=8)

        global get_image
        get_image = ImageTk.PhotoImage(Image.open(input_path))
        qr_label.configure(image=get_image, padx=20, pady=20)
        finished_label = ctk.CTkLabel(qr_frame, text="Finished", font=ctk.CTkFont(size=12))
        finished_label.pack(pady=10)
        entry.delete(0, ctk.END)
```



```
gui.py
gui.py > ...
        entry.delete(0, ctk.END)

def read_code():
    image_path = filedialog.askopenfilename(title="Select Image", filetypes=(("PNG files", "*.png"), ("All files", "*.*")))
    if image_path:
        image = cv2.imread(image_path)
        decoded_objects = pyzbar.decode(image)
        for obj in decoded_objects:
            decoded_text = obj.data.decode("utf-8")
            result_label = ctk.CTkLabel(scrollable_frame, text="" + decoded_text, font=ctk.CTkFont(size=12))
            result_label.pack(pady=5)
            return decoded_text
        result_label = ctk.CTkLabel(scrollable_frame, text="QR Code not found or could not be read", font=ctk.CTkFont(size=12))
        result_label.pack(pady=5)
        return "QR Code not found or could not be read"



title_label = ctk.CTkLabel(root, text="Generate and Read QR Code", font=ctk.CTkFont(size=24, weight="bold"))
title_label.pack(padx=10, pady=20)

scrollable_frame = ctk.CTkScrollableFrame(root, width=500, height=300)
scrollable_frame.pack()

entry = ctk.CTkEntry(scrollable_frame, placeholder_text="Add text or URL")
entry.pack(fill="x", pady=10)

add_button = ctk.CTkButton(root, text="Create Code", width=500, command=create_code)
add_button.pack(pady=10)

read_button = ctk.CTkButton(root, text="Read QR Code", width=500, command=read_code)
read_button.pack(pady=10)

qr_frame = ctk.CTkFrame(scrollable_frame)
qr_frame.pack(padx=10, pady=10)

qr_label = ctk.CTkLabel(qr_frame ,text="" )
qr_label.pack()


scrollable_frame = ctk.CTkScrollableFrame(root, width=500, height=10)
scrollable_frame.pack()

root.mainloop()
```

Code Features

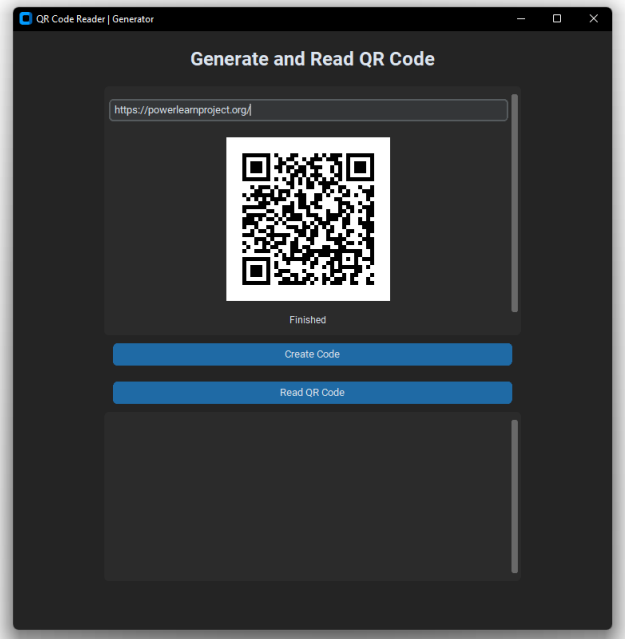
The QR Code Reader and Generator program is equipped with a wide range of features that make it a versatile tool for creating and reading QR codes. The program allows users to generate custom QR codes by inputting text or URLs, and save them as PNG files. Additionally, it provides options to adjust the scale of the generated code, and allows users to choose the file name and location for the saved image. The program also includes a built-in QR code reader that can decode QR codes from image files. This feature is useful for decoding QR codes that have been previously generated or shared by others. The program uses the PyZbar library to decode the QR code and display the decoded text in the program's GUI.

Another notable feature of the QR Code Reader and Generator program is its user-friendly interface. The program is built using the tkinter module, which provides a wide range of GUI components such as buttons, labels, and frames. These components are arranged in a logical and intuitive manner, making it easy for users to navigate the program's functionality. The program also includes a scrollable frame that allows users to view and interact with multiple components simultaneously. The scrollable frame provides a smooth and intuitive user experience, especially when dealing with long lists of decoded QR codes. Overall, the program's features and interface make it a valuable tool for generating and reading QR codes in a simple and user-friendly way.

Testing

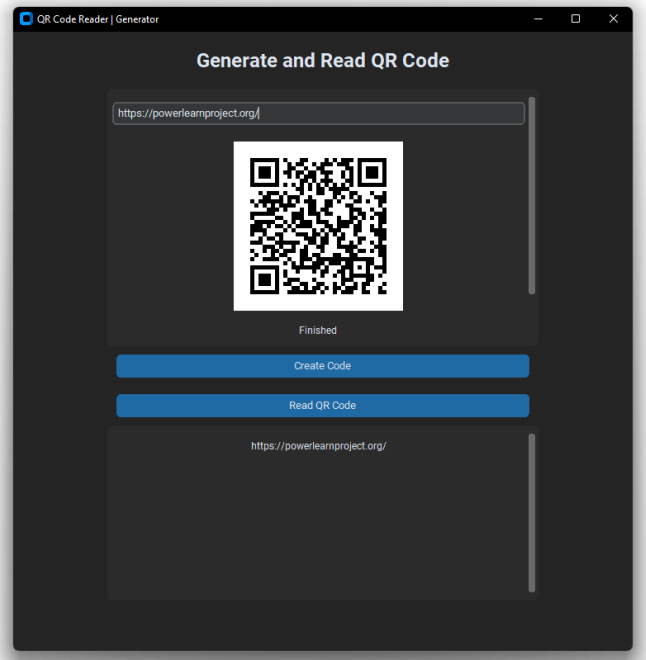
To test the program, you can enter text into the Entry widget and click the “Create Code” button to generate a QR code and display in the frame below.

Atlernatively, you can then click the “Read QR Code” button to select an image file containing a QR code and display the decoded text. If the QR code cannot be read, the program will display an error message.



Deployment

Deployment of this code can be done on any platform that supports Python and the necessary libraries, such as pyqrcode and pyzbar. The code can be executed from a command line interface or an IDE such as PyCharm. Once the code is deployed, the user can generate and read QR codes by inputting text or a URL. The GUI allows for easy interaction and a seamless user experience. The generated QR codes can be saved as PNG files, and the code also allows for the reading of QR codes from PNG files. The flexibility of deployment and ease of use make this code a useful tool for generating and reading QR codes.



Maintenance

maintenance is recommended. This includes keeping all necessary libraries and dependencies up to date, testing for any bugs or issues, and addressing any reported errors promptly. Additionally, incorporating user feedback and suggestions can enhance the user experience and improve the functionality of the code.

Conclusion

In conclusion, the QR code reader and generator is a useful tool for businesses and individuals alike. The code is structured in a clear and organized manner, making it easy for developers to understand and modify as needed. Its main features include the ability to generate and read QR codes, as well as the option to customize the size and format of the generated image.

In addition, the deployment process is straightforward and can be done on any machine that has the necessary libraries and modules installed. However, like any software, regular maintenance and updates are necessary to ensure its continued functionality and security. Overall, the QR code reader and generator is a valuable addition to any organization or individual looking for a simple and efficient way to generate and read QR codes.