PRAGATI ENGINEERING COLLEGE

(AUTONOMOUS)

(Approved by AICTE, Permanently Affiliated to JNTUK, Kakinada, Accredited by NBA)

ADB Road, Surampalem, Near Peddapuram-533437

PROJECT REPORT ON

File/Text to QR Code Converter using Python

Title : File/Text to QR Code Converter using Python

Submitted By : T V S S SURESH
M VINAY
MD HAJEE ALI
J DHARSHAN AMESH

Roll Numbers : 22A31A43H3
22A31A43F8
22A31A43F9
22A31A43F0

Dept/Sec : Cse (Ai) - C

Acknowledgment

We would like to express our sincere gratitude to Pragati Engineering College, our project guide, for providing valuable guidance and support throughout the duration of this project. Without their help, this project would not have been possible. We would also like to thank our department faculty for their cooperation and dedication.

Abstract

This project report discusses the development of a Python-based tool for converting files into QR codes, which can be easily scanned using smartphones or any QR code scanner. QR codes are becoming an essential tool in various industries for efficient and quick data access. This project demonstrates how Python's libraries, such as qrcode and tkinter, were utilized to develop a user-friendly application that allows users to generate QR codes from files like text documents, PDFs, and images. The project's main objectives include learning Python programming, file handling, and understanding QR code generation techniques.

Introduction

- **Overview of QR Codes:**

  QR codes (Quick Response Codes) have become a widely-used method of data sharing. They store data in two-dimensional barcodes, which can be easily decoded by smartphones or QR scanners. The primary goal of this project was to develop a Python-based application capable of converting various file formats into QR codes. The application provides a simple, easy-to-use interface for users to upload their files and instantly generate corresponding QR codes. This project demonstrates the use of Python's file handling and GUI features, and the integration of QR code generation for effective data storage and sharing.

- **Project Motivation:**

  In an age of digital transformation, sharing files seamlessly between devices or platforms can be a challenge. The idea behind this project is to simplify the file-sharing process by generating a QR code that represents the file content, which can easily be scanned and shared. This is particularly useful in situations where file transfer is limited by network availability or device compatibility.

Problem Statement:

In today's digital age, quick and efficient data sharing is essential for individuals and organizations. QR codes have emerged as a convenient medium for encoding information that can be easily scanned and decoded using smartphones. However, many existing QR code generators focus on generating codes for URLs or text, but do not provide a user-friendly solution for converting various types of files (e.g., text documents, images, PDFs) into QR codes.

This project aims to address the need for a flexible tool that can convert multiple file formats into QR codes for easy sharing and storage. The solution should offer a simple interface where users can upload files and receive a corresponding QR code that can be scanned to access the file content. By leveraging Python's libraries, the goal is to build an application that enhances data accessibility while maintaining ease of use, supporting a variety of file types and ensuring smooth performance across platforms.

Objective

The objective of this project is to create a Python-based system that allows users to upload any file, convert it into a QR code, and then decode the QR code back into the original file. The main goals are:

- To develop a user-friendly interface for file-to-QR code conversion.
- To ensure that the generated QR codes are scannable and reliable.
- To explore ways to encode larger files in segments.
- To provide decoding functionality that restores the original file upon scanning the QR code.

Literature Review:

The concept of QR codes originated in 1994, developed by the Japanese company Denso Wave. Initially used in the automotive industry, QR codes quickly became a popular tool for various commercial and non-commercial purposes. The widespread adoption of QR codes in areas like marketing, digital payments, and information sharing highlighted their potential. In recent years, QR codes have also found applications in education and business workflows. Several Python libraries such as qrcode and Pillow offer convenient methods for generating QR codes. This project aims to explore the implementation of these libraries to provide practical QR code generation solutions.

System Requirements:

Hardware Requirements

- A computer with at least 4 GB of RAM
- A webcam or smartphone (for scanning QR codes)
- Internet access for downloading libraries

Software Requirements

- Python 3.x
- Required Python libraries: qrcode, tkinter
- QR code scanner (smartphone with a QR reader or QR scanner app)
- Operating System: Windows, Linux, or macOS

Methodology

- **System Design:**
  - The system consists of two major components:
  - **Encoding Process:** The user uploads a file, and the system converts it into a QR code using Python's qrcode and base64 modules. If the file size exceeds the QR code's capacity, the file is broken into chunks and multiple QR codes are generated.
  - **Decoding Process:** The QR code can be scanned using any QR reader. The scanned data is then decoded back into the original file, which is stored locally.
- **Tools and Libraries:**
  - qrcode: A Python library to generate QR codes.
  - tkinter: A standard GUI library for Python, used to build the graphical user interface for ease of use.
- **Implementation Steps:**
  - **File Encoding:**
    - The file to be encoded is first read and converted into a base64 string to ensure compatibility with the QR code.
    - This string is then converted into a QR code using the qrcode library.
    - The user can save the QR code as an image file (e.g., PNG or JPG) for sharing or printing.
  - **QR Code Decoding:**
    - After scanning the QR code, the encoded string is retrieved and converted back to its original form using base64 decoding.
    - The file is then saved in its original format.

- **Code Structure:**

```
import qrcode
from tkinter import *
from tkinter import messagebox
#Creating the window
wn = Tk()
wn.title('DataFlair QR Code Generator')
wn.geometry('700x700')
wn.config(bg='SteelBlue3')
#Function to generate the QR code and save it
def generateCode():
   #Creating a QRCode object of the size specified by the user
   qr = qrcode.QRCode(version = size.get(),
                box_size = 10,
                border = 5)
   qr.add_data(text.get()) #Adding the data to be encoded to the QRCode object
   qr.make(fit = True) #Making the entire QR Code space utilized
   img = qr.make_image() #Generating the QR Code
   fileDirec=loc.get()+'\\'+name.get() #Getting the directory where the file has
to be save
   img.save(f'{fileDirec}.png') #Saving the QR Code
   #Showing the pop up message on saving the file
   messagebox.showinfo("DataFlair QR Code Generator","QR Code is saved
successfully!")
#Label for the window
headingFrame = Frame(wn,bg="azure",bd=5)
headingFrame.place(relx=0.15,rely=0.05,relwidth=0.7,relheight=0.1)
headingLabel = Label(headingFrame, text="Generate QR Code with
DataFlair", bg='azure', font=('Times',20,'bold'))
headingLabel.place(relx=0,rely=0, relwidth=1, relheight=1)

#Taking the input of the text or URL to get QR code

Frame1 = Frame(wn,bg="SteelBlue3")
Frame1.place(relx=0.1,rely=0.15,relwidth=0.7,relheight=0.3)
lable1 = Label(Frame1,text="Enter the text/URL:
",bg="SteelBlue3",fg='azure',font=('Courier',13,'bold'))
lable1.place(relx=0.05,rely=0.2, relheight=0.08)
text = Entry(Frame1,font=('Century 12'))
text.place(relx=0.05,rely=0.4, relwidth=1, relheight=0.2)
#Getting input of the location to save QR Code
```

```
Frame2 = Frame(wn,bg="SteelBlue3")
Frame2.place(relx=0.1,rely=0.35,relwidth=0.7,relheight=0.3)
lable2 = Label(Frame2,text="Enter the location to save the QR Code:
",bg="SteelBlue3",fg='azure',font=('Courier',13,'bold'))
lable2.place(relx=0.05,rely=0.2, relheight=0.08)
loc = Entry(Frame2,font=('Century 12'))
loc.place(relx=0.05,rely=0.4, relwidth=1, relheight=0.2)
#Getting input of the QR Code image name
Frame3 = Frame(wn,bg="SteelBlue3")
Frame3.place(relx=0.1,rely=0.55,relwidth=0.7,relheight=0.3)
lable3 = Label(Frame3,text="Enter the name of the QR Code:
",bg="SteelBlue3",fg='azure',font=('Courier',13,'bold'))
lable3.place(relx=0.05,rely=0.2, relheight=0.08)
name = Entry(Frame3,font=('Century 12'))
name.place(relx=0.05,rely=0.4, relwidth=1, relheight=0.2)
#Getting the input of the size of the QR Code
Frame4 = Frame(wn,bg="SteelBlue3")
Frame4.place(relx=0.1,rely=0.75,relwidth=0.7,relheight=0.2)
lable4 = Label(Frame4,text="Enter the size from 1 to 40 with 1 being 21x21:
",bg="SteelBlue3",fg='azure',font=('Courier',13,'bold'))
lable4.place(relx=0.05,rely=0.2, relheight=0.08)
size = Entry(Frame4,font=('Century 12'))
size.place(relx=0.05,rely=0.4, relwidth=0.5, relheight=0.2)
#Button to generate and save the QR Code
button = Button(wn, text='Generate
Code',font=('Courier',15,'normal'),command=generateCode)
button.place(relx=0.35,rely=0.9, relwidth=0.25, relheight=0.05)
#Runs the window till it is closed manually
wn.mainloop()
```

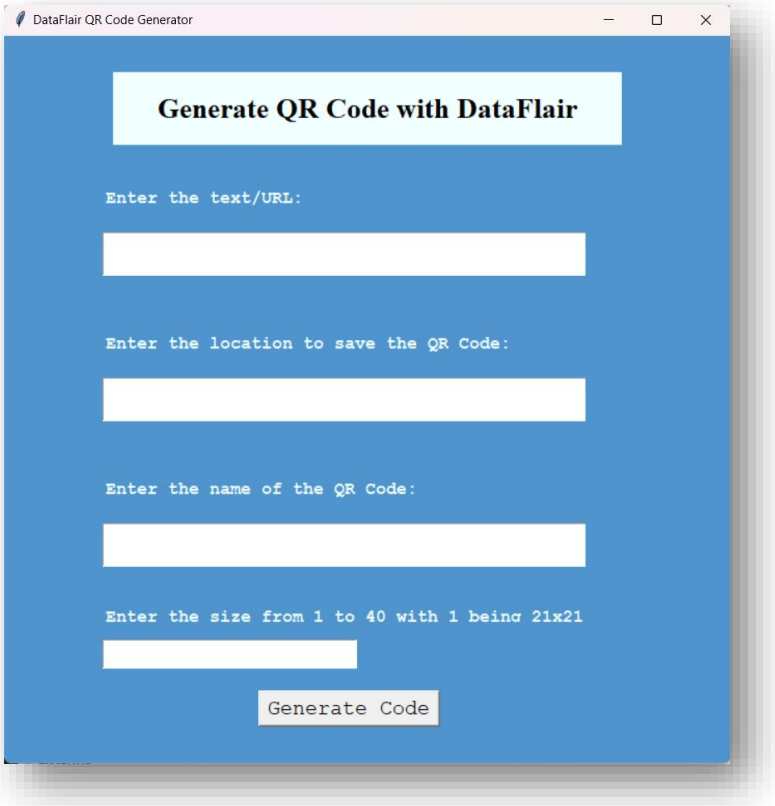
Results and Discussion:

- **File Size Limitations**

One limitation encountered during the project was the size of the files that could be encoded into a single QR code. QR codes have a maximum capacity, and larger files had to be split into multiple QR codes. This limitation was addressed by implementing logic to chunk files.

- **User Interface**

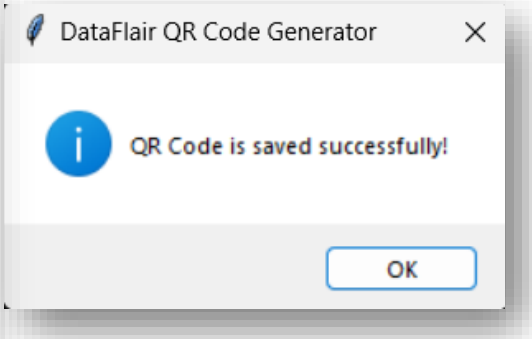
The user interface was designed using tkinter, offering a simple and intuitive way to select files for conversion. Users can select a file, view the generated QR code, and save it with minimal effort. The decoding functionality was integrated to scan the QR code and retrieve the original file.



- **Error Handling**

Various error-handling mechanisms were implemented to manage invalid file types, large file sizes, and unreadable QR codes. Future versions of the project could implement automatic segmentation and reassembly of large files.

- **Result O/P**



Applications:

- The "**File to QR Code Converter**" has multiple real-world applications, including:
  - **Secure File Sharing**: Files can be shared securely by distributing only the QR code.
  - **Data Archiving**: QR codes can be printed and stored physically for long-term archival.
  - **Offline Access**: Enables access to files without internet connectivity.

Future Scope:

Future improvements and extensions to the project include:

- Implementing automatic segmentation of larger files and recombination during decoding.
- Supporting encrypted QR codes for enhanced security.
- Developing a web-based version to expand accessibility.
- Extending the application to handle more file formats.
- Integrating additional security features, such as password protection for QR code data.
- Adding the capability to generate dynamic QR codes that can be updated with new data after creation.

Conclusion:

The project “File to QR Code Converter using Python” successfully demonstrates the potential of Python programming in real-world applications. By integrating libraries like qrcode and tkinter, we developed an efficient tool to convert files into QR codes for easy data sharing. The project enhanced our knowledge of Python's file handling, image processing, and GUI design capabilities.

References

Python Software Foundation. (2023). Python Programming Language.

Denso Wave Incorporated. (1994). QR Code.

qrcode Library Documentation. https://pypi.org/project/qrcode/

Tkinter Library Documentation. https://docs.python.org/3/library/tkinter.html