Report On

QR Code Generator

Submitted in partial fulfillment of the requirements of the Course project in Semester IV of Second Year Computer Engineering

By

Ayush Sahu(25)

Vaibhav Sapaliya(28)

Aishwarya Shinde(39)

Supervisor Ms. Sneha Mhatre

## Vidyavardhini's College of Engineering & Technology Department of Computer Engineering



**(2023-24)**

# Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

**CERTIFICATE**

This is to certify that the project entitled “QR Code Generator” is a bonafide work of “Ayush Sahu(25), Vaibhav Sapaliya(28),Aishwarya Shinde(39)”submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Computer Engineering.

Ms. Sneha Mhatre Mentor

|  |  |
| --- | --- |
| Dr Megha Trivedi  Head of Department | Dr. H.V. Vankudre  Principal |

# ABSTRACT

QR (Quick Response) codes have become ubiquitous in our digital world, serving as a quick and efficient method for transferring information. With the increasing reliance on smartphones and digital devices, QR codes offer a convenient way to access websites, share contact details, make payments, and much more. In this project, we propose a QR code generator that allows users to easily create customized QR codes for various purposes.

The QR code generator will be a web-based application accessible from any device with an internet connection. Users will have the option to input different types of data, including URLs, text, contact information, Wi-Fi credentials, and more. The generator will then encode this data into a QR code, which users can download or share directly from the application.

# CONTENTS:

## Pg. No

|  |  |
| --- | --- |
| 1.Report Page | 1 |
| 2. Certificate | 2 |
| 3.Abstract | 3 |
| 5.Module Description 6.Software Required 7.Program   1. Result 2. Conclusion | 5  8  9  10  11 |

**MODULE DESCRIPTION AND FLOWCHART:**

* Module Overview: The QR Code Generator module provides functionality to generate QR (Quick Response) codes dynamically based on user input. QR codes are two-dimensional barcodes that encode various types of data, such as URLs, text, contact information, and more. This module enables users to create customized QR codes for different purposes, including website links, contact details, Wi-Fi credentials, and product information.
* Key Features:
* Dynamic QR Code Generation: Users can input different types of data, and the module generates a corresponding QR code image in real-time.
* Customization Options: Users can customize the appearance of the QR code by selecting colors, adding logos or icons, and adjusting error correction levels.
* Multi-Format Support: The module supports encoding various data formats, including URLs, plain text, phone numbers, email addresses, Wi-Fi credentials (SSID and password), and vCard/contact information.
* Batch Generation: Users can generate multiple QR codes in bulk, either by uploading a list of data or by specifying multiple inputs simultaneously.
* Download and Sharing: Users can download the generated QR code image in different formats (e.g., PNG, JPEG) or share it directly via email, social media, or messaging platforms.
* Analytics Tracking (Optional): An optional feature allows users to track QR code scans and gather analytics data, such as scan location, time, and device information.
* API Integration: The module provides an API for developers to integrate QR code generation functionality into their own applications or websites seamlessly.
* Module Components:
* User Interface (UI): The UI component provides a user-friendly interface for users to input data, customize QR code settings, and download or share generated QR codes.
* QR Code Generation Engine: The core component responsible for encoding user input into QR code images using standard QR code generation algorithms and libraries.
* Customization Module: Allows users to customize the appearance of QR codes, including color selection, logo embedding, and error correction level adjustment.
* Batch Processing Module: Handles bulk QR code generation tasks, such as processing multiple inputs, generating QR codes in batches, and managing download or sharing options.
* Analytics Module (Optional): Implements analytics tracking functionality to monitor QR code scans and collect relevant data for analysis and reporting purposes.
* API Module: Provides a RESTful API interface for developers to integrate QR code generation functionality into third-party applications or websites programmatically.

## SOFTWARE REQUIRED:

Development Environment:

1. Python: The core language used for the project. Ensure you have the latest Python version installed, or at least a version that supports all the libraries you plan to use.
2. Integrated Development Environment (IDE):
   * PyCharm: Highly recommended for Python projects due to its robust features, including code completion, project management, and debugging tools.
   * Visual Studio Code: A lightweight, versatile editor with strong Python support via extensions.

Libraries Module:

1. **tkinter**: Used for creating a graphical user interface (GUI) window. It provides various widgets and tools for building interactive applications with buttons, labels, entry fields, etc.
2. **qrcode**: Used for generating QR codes. This library allows the program to encode data into a QR code format, such as URLs, text, contact information, etc.
3. **PIL (Python Imaging Library)**: Used for working with images. In this script, it is specifically used for displaying images in the GUI window. The **Image** module is used to open and manipulate image files, while **ImageTk** is used for displaying images in tkinter windows.
4. **resizeimage**: Used for resizing images. This library provides functionality to resize images to specific dimensions, which may be useful for adjusting the size of the QR code image before displaying it in the GUI.

## PROGRAM:

new1.py

from logging import root

from tkinter import \* #----------tkinter use for creating an interferance window-------------

import qrcode as qr #----------which will create qr code----------------

from PIL import Image,ImageTk #------------will convert in image form--------------

from resizeimage import resizeimage #---------use for resizing the image----------------

class Qr\_Generator: #------------class has been used------------------

def \_init\_(self, root):

self.root=root

self.root.geometry("900x500+200+50") #-----------dimension for window------------

self.root.title("QR CODE GENERATOR || By Atharv Sawant")

self.root.resizable(False,False) #---------it will not maximize the window------------

#self.root.configure(bg="#d8bfd8")

self.root.configure(bg="white")

title=Label(self.root,text="Qr Code Generator",font=("times new roman",40),bg="#CE71AF",fg="#89138A").place(x=0,y=0,relwidth=1)

self.var\_bk\_name=StringVar()

self.var\_bk\_author=StringVar()

self.var\_bk\_link=StringVar()

#self.var\_bk\_id=StringVar()

#self.var\_bk\_med=StringVar()

#-------------Book Information Window---------------

bk\_frame=Frame(self.root,bd=2,relief=RIDGE,bg="white")

bk\_frame.place(x=50,y=100,width=500,height=380)

bk\_title=Label(bk\_frame,text="BOOK DETAILS",font=("goudy old style",30),bg="#CE71AF",fg="#89138A").place(x=0,y=0,relwidth=1)

lbl\_bk\_name=Label(bk\_frame,text="BOOK NAME",font=("times new roman",15),bg="white",fg="black").place(x=20,y=70)

lbl\_bk\_author=Label(bk\_frame,text="AUTHOR NAME",font=("times new roman",15),bg="white",fg="black").place(x=20,y=110)

lbl\_bk\_link=Label(bk\_frame,text="LINK",font=("times new roman",15),bg="white",fg="black").place(x=20,y=150)

txt\_bk\_name=Entry(bk\_frame,font=("times new roman",15),textvariable=self.var\_bk\_name,bg="light yellow",fg="black").place(x=250,y=70)

txt\_bk\_author=Entry(bk\_frame,font=("times new roman",15),textvariable=self.var\_bk\_author,bg="light yellow",fg="black").place(x=250,y=110)

txt\_bk\_link=Entry(bk\_frame,font=("times new roman",15),textvariable=self.var\_bk\_link,bg="light yellow",fg="black").place(x=250,y=150)

#-----------------buttons functionality----------------------

btn\_generator=Button(bk\_frame,text="Generate QR",command=self.generate,font=("times new roman",18,"bold"),bg="#CE71AF",fg="#89138A").place(x=60,y=210,width=200,height=30)

btn\_reset=Button(bk\_frame,text="Reset",command=self.Reset,font=("times new roman",18,"bold"),bg="#CE71AF",fg="#89138A").place(x=300,y=210,width=150,height=30)

btn\_Save=Button(bk\_frame,text="Save QR",command=self.save,font=("times new roman",18,"bold"),bg="#CE71AF",fg="#89138A").place(x=190,y=270,width=150,height=30)

#--------------Message Display Screen-----------------------

self.msg="Please Enter the detalis for QR Code"

self.lbl\_msg=Label(bk\_frame,text=self.msg,font=("times new roman",20),bg="white",fg="green")

self.lbl\_msg.place(x=0,y=320,relwidth=1)

#---------------Book QR Code Window------------------

qr\_frame=Frame(self.root,bd=2,relief=RIDGE,bg="white")

qr\_frame.place(x=600,y=100,width=250,height=380)

bk\_title=Label(qr\_frame,text="BOOK QR",font=("goudy old style",30),bg="#CE71AF",fg="#89138A").place(x=0,y=0,relwidth=1)

self.qr\_code=Label(qr\_frame,text="QR Code \nNot Available",font=("times new roman",20),bg="#CE71AF",fg="white",bd=1,relief=RIDGE)

self.qr\_code.place(x=40,y=100,width=180,height=180)

def Reset(self): #-------------Object for rest button-----------------

self.var\_bk\_name.set("")

self.var\_bk\_author.set("")

self.var\_bk\_link.set("")

#self.var\_bk\_med.set("")

self.msg="Please Enter the detalis for QR Code"

self.lbl\_msg.configure(text=self.msg,fg="green")

self.qr\_code.config(image="")

def generate(self): #-----------------Object for generate qr code button--------------

if self.var\_bk\_name.get()=="" or self.var\_bk\_author.get()=="" or self.var\_bk\_link.get()=="":

self.msg="All Fields are Required"

self.lbl\_msg.configure(text=self.msg,fg="red")

else:

qr\_data=(f"link:- {self.var\_bk\_link.get()}") #----------use of F string---------- syntax = name {variable.get()}

qr\_code=qr.make(qr\_data) #--------------this function will generate qr code--------------

qr\_code=resizeimage.resize\_cover(qr\_code,[180,180]) #----------will resize the image----------

self.im=ImageTk.PhotoImage(qr\_code) #---------the image will be add in the tkwinter -----------

self.qr\_code.config(image=self.im)

self.msg="QR Code Generated Sucessfully"

self.lbl\_msg.configure(text=self.msg,fg="green")

def save(self): #-----------Object for Save the qr code----------------

if self.var\_bk\_name.get()=="" or self.var\_bk\_author.get()=="" or self.var\_bk\_link.get()=="":

self.msg="All Fields are Required"

self.lbl\_msg.configure(text=self.msg,fg="red")

else:

qr\_data=(f"link:- {self.var\_bk\_link.get()}")

qr\_code=qr.make(qr\_data)

qr\_code.save("BK\_"+str(self.var\_bk\_name.get())+"AU"+str(self.var\_bk\_author.get())+'.png')

qr\_code=resizeimage.resize\_cover(qr\_code,[180,180])

self.im=ImageTk.PhotoImage(qr\_code)

self.qr\_code.config(image=self.im)

self.msg="QR Code Saved Sucessfully "

self.lbl\_msg.configure(text=self.msg,fg="green")

root=Tk()

obj = Qr\_Generator(root)

root.mainloop(

## RESULTS:

****

**CONCLUSION**:

In conclusion, the QR Code Generator module offers a versatile and user-friendly solution for generating customized QR codes to meet diverse needs. Throughout this project, we have developed a robust system that empowers users to create QR codes dynamically based on their input data, whether it be URLs, text, contact information, or other formats.With a focus on user experience and functionality, our module provides a range of features to enhance the QR code generation process. Users can easily customize the appearance of QR codes by selecting colors, adding logos or icons, and adjusting error correction levels to ensure optimal readability and visual appeal.Additionally, our module supports batch generation, allowing users to create multiple QR codes efficiently, whether for personal or business use. The ability to download or share generated QR codes in various formats further enhances the flexibility and accessibility of our solution.