

QR CODE GENERATOR

Mini Project Report submitted by

1. CHETHANA R KINI 4NM21AI018	2. MALLIKARJUNA N A 4NM21AI037
3.MANYA HEGDE 4NM21AI038	

Under the Guidance of

Mr. Sudesh Rao

Assistant Professor Department of Artificial Intelligence and Machine
Learning Engineering

In partial fulfillment of the requirements for the

Object Oriented Programming in Python (21AM305)

**Bachelor of Engineering in Artificial Intelligence and Machine
Learning Engineering**

from

NMAM INSTITUTE OF TECHNOLOGY, NITTE

Department of Artificial Intelligence and Machine Learning Engineering

NMAM Institute of Technology, Nitte - 574110

(An Autonomous Institution under VTU, Belagavi)



NITTE
EDUCATION TRUST

**NMAM INSTITUTE
OF TECHNOLOGY**



NITTE
EDUCATION TRUST

**NMAM INSTITUTE
OF TECHNOLOGY**

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Certified that the mini project work entitled

“QR CODE GENERATOR”

is a bonafide work carried out by as a component of

Object Oriented Programming in Python – 21AM305

CHETHANA R KINI - 4NM21AI018

MALLIKARJUNA N A - 4NM21AI037

MANYA HEGDE – 4NM21AI038

in partial fulfilment of the requirements for the award of

***Bachelor of Engineering Degree in Artificial Intelligence and Machine Learning
Engineering***

prescribed by NMAM Institute of Technology, Nitte

during the year 2022-2023.

*It is certified that all corrections/suggestions indicated for Internal Assessment have been
incorporated in the report deposited in the departmental library.*

*The mini project report has been approved as it satisfies the academic requirements in
respect of the*

mini project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide

Signature of HOD

EVALUATION

Name of the Examiners

Signature with Date

1. _____

2. _____

ACKNOWLEDGEMENT

We believe that our mini project will be complete only after we thank the people who have contributed to make this mini project successful.

First and foremost, our sincere thanks to our beloved principal, Dr. Niranjan N. Chiplunkar for giving us an opportunity to carry out our mini project work at our college and providing us with all the needed facilities.

We acknowledge the support and valuable inputs given by, Dr. Sharada U Shenoy the Head of the Department, Artificial Intelligence and Machine Learning Engineering, NMAMIT, Nitte.

We express our deep sense of gratitude and indebtedness to our guide Mr. Sudesh Rao, Assistant Professor Artificial Intelligence and Machine Learning Engineering, for his inspiring guidance, constant encouragement, support and suggestions for improvement during the course for our mini project.

We also thank all those who have supported us throughout the entire duration of our mini project.

Finally, we thank the staff members of the Department of Artificial Intelligence and Machine Learning Engineering and all our friends for their honest opinions and suggestions throughout the course of our mini project.

Chethana R Kini
Mallikarjuna N A
Manya Hegde

ABSTRACT

Quick Response (QR) codes seem to appear everywhere these days. We can see them on posters, magazine ads, websites, product packaging and so on. Using the QR codes is one of the most intriguing ways of digitally connecting consumers to the internet via mobile phones since the mobile phones have become a basic necessity thing of everyone. In this paper, we present a methodology for creating QR codes by which the users enter link and get the QR code generated.

Quick Response law is a 2D matrix law that's designed by keeping two points under consideration, i.e. it must store large quantities of data as compared to 1D barcodes and it must be decrypted at high speed using any handheld device like phones. QR law provides high data storehouse capacity, fast scanning, omnidirectional readability, and numerous other advantages including, error-correction (so that damaged law can also be read successfully) and different types of performances. Different kinds of QR law symbols like totem QR law, translated QR law, iQR Code are also available so that stoner can choose among them according to their need. currently, a QR law is applied in different operation aqueducts related to marketing, security, academicsetc. and gain fashionability at a really high pace. Day by day more people are getting apprehensive of this technology and use it consequently. The fashionability of QR law grows fleetly with the growth of smartphone druggies and therefore the QR law is fleetly arriving at high situations of acceptance worldwide.

The fashionability of QR code grows fleetly with the growth of smartphone druggies and therefore the QR law is hastily arriving at high situations of acceptance worldwide. With the wide perpetration of QR law, the protection point of QR law is serious, like data leakage and data revision. This paper emphasizes on the analysis of QR law and its applications. This platform could be used by different security heart associations. Text lines or word system could be translated into QR Code and be read by a mobile device, etc. The work is achieved by the use of python beaker frame which is the main interface for generating the QR Canons.

Identification of Problem

A QR code uses four more advanced error correction mechanism and are more standardized encoding modes (numeric, alphanumeric, reliable as it has a faster speed than other codes (Adeel et al., byte/binary, and kanji) to efficiently store data; extensions 2014). It has a larger set of machine readable instructions.

TABLE OF CONTENTS:

DESCRIPTION			PAGE NO.
Acknowledgement			3
Abstract			4
1.	Introduction		
	1.1.	Overview	6
	1.2.	Objectives	6
2.	Graphical User Interface		
	2.1.	GUI	7
	2.2	Tkinter	
		2.2.1.	Tkinter
		2.2.2.	Tkinter widgets
		2.2.3.	User-defined functions
		2.2.4.	Geometry Management
3.	Python modules		
	3.1.	Overview	9
	3.2.	The import statement	10
4.	QR code python		
	4.1	QR code	11
	4.2	Uses and importance of QR Codes	13
	4.3	Steps to build the QR Code Generator in Python	13
5.	Source Code		16
6.	Result		19
7.	System requirements		21
8.	Conclusion		21
9.	References		22

1. INTRODUCTION

1.1. Overview

QR stands for "Quick Response." While they may look simple, QR codes are capable of storing lots of data. But no matter how much they contain, when scanned, the QR code should allow the user to access information instantly – hence why it's called a Quick Response code.

QR Code is two-dimensional barcode which is categorized in matrix barcode that can store data information. QR stands for "Quick Response" as the creator intended the code to allow its contents to be decoded at high speed. It is introduced in Japan by Denso Corporation in 1994. This kind of barcode was initially used for tracking inventory in vehicle parts manufacturing and is now used in a variety of industries. Nowadays, mobile phones with built-in camera are widely used to recognize the QR code. There have been many URL shortening services that automatically generate QR code links to websites such as Goo.gl and Bit.ly. Goo.gl is the first introduced URL shortening service that provides automatically generates QR codes. only 12 days after introducing Goo.gl, Bit.ly launched the same service. The URL shortening service shortens link and turn it into a QR code that, when scanned with a mobile QR code reader, automatically direct users to the shortened link. It really shows that QR codes are going to become more and more popular.

Which algorithm is used to generate a QR code?

The Reed Solomon Correction Method:

This is all thanks to the Reed Solomon Correction Method. Basically, the Reed Solomon method is an algorithm that all QR code readers have built-in as standard. It allows QR codes to be scanned even if a certain amount of the QR code is covered up or blocked.

1.2. Objectives

In this mini project, we aim to create a QR code Generator that can be used to generate a QR code of any website, apps etc. Thus, the main objective of this project are :

- To generate a code just by pasting the link or by typing the link.
- To allow users to generate the qr code of their desired size.
- To download the qr code generated.
- Saves the time of users and easy to access.

2. GRAPHICAL USER INTERFACE

2.1. GUI:

Graphical User Interface (GUI) is a form of user interface which allows users to interact with computers through visual indicators using items such as icons, menus, windows, etc. It has advantages over the Command Line Interface (CLI) where users interact with computers by writing commands using keyboard only and whose usage is more difficult than GUI. Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run. Although Tkinter is considered the de-facto Python GUI framework, it's not without criticism. One notable criticism is that GUIs built with Tkinter look outdated. If you want a shiny, modern interface, then Tkinter may not be what you're looking for.

However, Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to build something that's functional and cross-platform quickly.

Python provides various options for developing graphical user interfaces (GUIs) like Tkinter.

2.2. Tkinter :

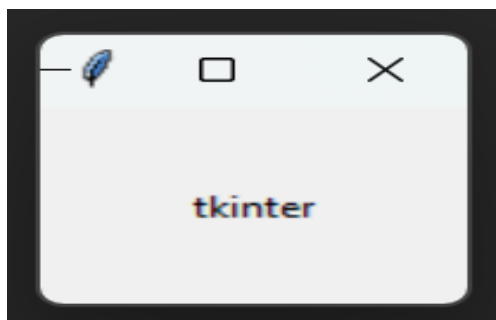
2.2.1. Tkinter

Tkinter is a graphical user interface (GUI) module for Python, you can make desktop apps with Python. You can make windows, buttons, show text and images amongst other things. Tk and Tkinter apps can run on most Unix platforms. This also works on Windows and Mac OS X. The module Tkinter is an interface to the Tk GUI toolkit.

For example

```
In [14]: from tkinter import *
         window=Tk()
         l1=Label(window,text="tkinter",height=5)
         l1.pack()
         window.mainloop()
         |
```

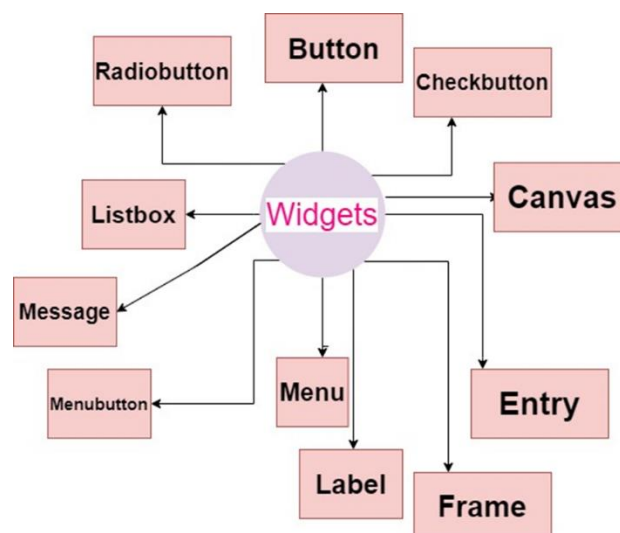
Output:



2.2.2. Tkinter widgets:

Tkinter provides various controls, such as buttons, labels and text boxes used in GUI application. These controls are commonly called as widgets.

Widgets	Description
Label	It is used to display text or image on the screen
Button	It is used to add buttons to your application.
Canvas	It is used to draw pictures and others layouts like texts, graphics etc.
ComboBox	It contains a down arrow to select from list of available options.
CheckButton	It displays a number of options to the user as toggle buttons from which user can select any number of options.
Radio Button	It is used to implement one-of-many selection as it allows only one option to be selected.
Entry	It is used to input single line text entry from user.
Frame	It is used as container to hold and organize the widgets.
Message	It works same as that of label and refers to multi-line and non-editable text.
Scale	It is used to provide a graphical slider which allows to select any value from that scale.
Scrollbar	It is used to scroll down the contents. It provides a slide controller.
SpinBox	It is allows user to select from given set of values.
Text	It allows user to edit multiline text and format the way it has to be displayed.
Menu	It is used to create all kinds of menu used by an application.



User-defined functions:

- generate_QR() where all our QR code logic will reside.
- The pyqrcode.create() generates the QR code as it fetches the string through user_input.get() method from the text box.
- Also, this function will get executed when the user_input.get() is not equal to 0.
- Once the qr code gets generated using the pyqrcode.create(), we have to use the BitmapImage() and pass the data = qr.xbm()) along with a scale size (here 10) that will generate a Bitmap image of 10x10.
- If the user_input.get() is equals to zero, messagebox.showwarning() shows a warning to fill the data in that text box. Then within the try block we call the display_code().
- the mainloop() is used, which is an infinite loop implemented for running the application, wait for an event to occur and process the event as long as the window does not get closed by the user manually.

2.2.3. Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The **pack()** Method - This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The **grid()** Method - This geometry manager organizes widgets in a table-like structure in the parent widget.
- The **place()** Method - This geometry manager organizes widgets by placing them in a specific position in the parent widget.

3. Python Modules

3.1. Overview:

There are actually three different ways to define a module in Python:

1. A module can be written in Python itself.
2. A module can be written in C and loaded dynamically at run- time, like the re (regular expression) module.
3. A built-in module is intrinsically contained in the interpreter, like the itertools module.

A module's contents are accessed the same way in all three cases: with the import statement

Here, the focus will mostly be on modules that are written in Python. The cool thing about modules written in Python is that they are exceedingly straightforward to build. All you need to do is create a file that contains legitimate Python code and then give the file a name with a.py extension. That's it! No special syntax or voodoo is necessary.

For example, suppose you have created a file called mod.py containing the following:

```
mod.py
s="If Comrade Napoleon says it, it must be right."
a = [100, 200, 300]
def foo(arg):
    print (f'arg= {arg}')
class Foo:
    pass
```

Several objects are defined in mod.py:

```
s (a string)
a (a list)
foo() (a function)
Foo (a class)
```

Assuming mod.py is in an appropriate location, which you will learn more about shortly, these objects can be accessed by importing the module as follows:

```
>>> import mod
>>> print(mod.s) If Comrade Napoleon says it, it must be right. >>> mod.
[100, 200, 300]
>>> mod. foo(['quux', 'corge', 'gault'])
arg= ['quux', 'corge', 'gault'] >>> x = mod. Foo() >>> X
<mod. Foo object at 0x03C181F0>
```

3.2.The import Statement:

Module contents are made available to the caller with the import statement. The import statement takes many different forms, shown below.

Import <module_name>

The simplest form is the one already shown above:

```
import <module_name>
```

that this does not make the module contents directly accessible to the caller. Each module has its own private symbol table, which Note serves as the global symbol table for all objects defined in the module. Thus, a module creates a separate namespace, as already noted.

statement import

The places <module_name> in the caller's symbol table. The objects that <module_name> only are defined in the module remain in the module's private symbol table.

From the caller, objects in the module are only accessible when prefixed with <module_name> via dot notation, as illustrated below. After the following import statement, mod is placed into the local

symbol table. Thus, mod has meaning in the caller's local context:

```
>>> import mod >>> mod
```

module 'mod' from

```
"C:\\Users\\John\\Documents\\Python\\doc\\mod.py">
```

But s and foo remain in the module's private symbol table and are not meaningful in the local context:

```
>>> 5
```

```
NameError: name 's' is not defined >>> foo('quux')
```

```
NameError: name 'foo' is not defined
```

To be accessed in the local context, names of objects defined in the module must be prefixed by mod:

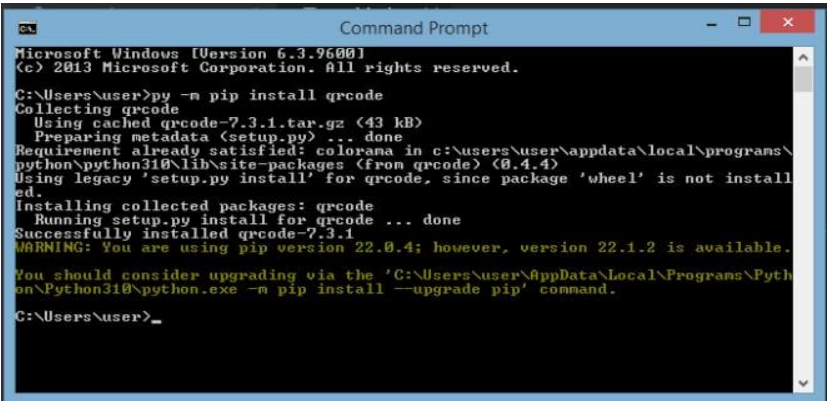
```
>>> mod.s
```

```
'If Comrade Napoleon says it, it must be right.'
```

```
>>> mod.foo('quux') arg quux
```

Several comma-separated modules may be specified in a single import statement:

```
import <module_name> [, <module_name> ...]
```



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\user>py -m pip install qr
Collecting qr
  Using cached qr-7.3.1.tar.gz (43 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: colorama in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (from qr) (0.4.4)
Using legacy 'setup.py install' for qr, since package 'wheel' is not installed.
Installing collected packages: qr
  Running setup.py install for qr ... done
Successfully installed qr-7.3.1
WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the 'C:\Users\user\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip' command.

C:\Users\user>
```

4. QR code python:

4.1. QR code

A Quick Response Code or a QR Code is a two-dimensional bar code used for its fast readability and comparatively large storage capacity. It consists of black squares arranged in a

square grid on a white background. Python has a library “qrcode” for generating QR code images. It can be installed using pip.



The following softwares can be used to create a qrcode:

i Jupyter notebook:

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

ii Thonny:

Thonny is a free development program for PC that was made by an independent dev who goes by the same name. It is an open-source integrated development environment (IDE) that can be used to create various applications using the Python programming language.

Just like Microsoft Visual Studio or NetBeans IDE, Thonny makes it easier for programmers to code as it already comes with the essential tools, libraries, and dependencies that they need to get started. This particular IDE was made to focus on Python and to cater to beginners who want to learn to code and make programs with it.

iii PyCharm:

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development

iv OpenCV:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

4.2. Uses and importance of QR Codes:

Some of the major uses of QR Codes in the world as we see today include:

- In online payment to access account numbers or pay bills
- Advertisements to land on webpages
- By companies to advertise and download apps
- Shopping and Ecommerce
- Direct clients and potential customers on social media platforms and so much more

4.3. Steps to build the QR Code Generator in Python:

To build the QR code generator project using Python we need to follow the below steps:

- 1.Importing the modules
2. Creating the main window
3. Taking the input of the text/URL, location to store the QR code, name of the QR code and the size of the QR code
4. Writing the function to generate and save the QR Code

1. Importing the modules

The first step is to import the qrcode and the tkinter module. We use the messagebox in the tkinter module to show the pop up messages.

```

1. #Creating the window
2. wn = Tk()
3. wn.title('DataFlair QR Code Generator')
4. wn.geometry('700x700')
5. wn.config(bg='SteelBlue3')

```

2. Creating the main window

Next, we create the main window with title, size and color.

```

1. #Creating the window
2. wn = Tk()
3. wn.title('DataFlair QR Code Generator')
4. wn.geometry('700x700')
5. wn.config(bg='SteelBlue3')

```

3. Taking the inputs and creating the function to generate QR code and save it

Now, we take the inputs from the user to create the QR Code. We take the following inputs:

1. Text/URL as Entry() named as 'text'.
2. Location to save the QR Code as Entry() named as 'loc'.
3. Name of the QR Code image when saved in the device as Entry() named as 'name'.
4. Size of the QR Code to be generated as Entry() named 'size'. In this the user has to give the size in the range 1-40. 1 being the smallest size of 21×21. Then we create a button when clicked generates the QR Code and saves it by executing the generateCode() function.
5. Then we create the QRCode object with the version/size that user gave as input in the size() entry.
6. Then we add the text that we need to encode by getting from the entry 'text'.
7. Then we get the QR code and save it in the directory that user gave as input.
8. After this, we show the pop up message to confirm the user that the image is saved.

```

import qrcode
from tkinter import *
from tkinter import messagebox
#Creating the window
wn = Tk()
wn.title('QR CODE GENERATOR')
wn.geometry('700x700')
wn.config(bg='Black')

#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("QR Code Generator","QR Code is saved Successfully!!")

#Label for the window
headingFrame = Frame(wn,bg="azure",bd=5)
headingFrame.place(relx=0.15,relly=0.05,relwidth=0.7,relheight=0.1)
headingLabel = Label(headingFrame, text="Generate QR Code with DataFlair", bg='Yellow', font=('Times',20,'bold'))
headingLabel.place(relx=0,relly=0, relwidth=1, relheight=1)

#Taking the input of the text or URL to get QR code
Frame1 = Frame(wn,bg="Black")
Frame1.place(relx=0.1,relly=0.15,relwidth=0.7,relheight=0.3)

label1 = Label(Frame1,text="Enter the text/URL: ",bg="Black",fg='azure', font=('Courier',13,'bold'))
label1.place(relx=0.05,relly=0.2, relheight=0.08)

text = Entry(Frame1,font=('Century 12'))
text.place(relx=0.05,relly=0.4, relwidth=1, relheight=0.2)

#Getting input of the location to save QR Code
Frame2 = Frame(wn,bg="Black")
Frame2.place(relx=0.1,relly=0.35,relwidth=0.7,relheight=0.3)

label2 = Label(Frame2,text="Enter the location to save the QR Code: ",bg="Black",fg='azure',font=('Courier',13,'bold'))
label2.place(relx=0.05,relly=0.2, relheight=0.08)

loc = Entry(Frame2,font=('Century 12'))
loc.place(relx=0.05,relly=0.4, relwidth=1, relheight=0.2)

#Getting input of the QR Code image name
Frame3 = Frame(wn,bg="Black")
Frame3.place(relx=0.1,relly=0.55,relwidth=0.7,relheight=0.3)

label3 = Label(Frame3,text="Enter the name of the QR Code: ",bg="Black",fg='azure',font=('Courier',13,'bold'))
label3.place(relx=0.05,relly=0.2, relheight=0.08)

name = Entry(Frame3,font=('Century 12'))
name.place(relx=0.05,relly=0.4, relwidth=1, relheight=0.2)

#Getting the input of the size of the QR Code
Frame4 = Frame(wn,bg="Black")
Frame4.place(relx=0.1,relly=0.75,relwidth=0.7,relheight=0.2)

label4 = Label(Frame4,text="Enter the size from 1 to 40 with 1 being 21x21: ",bg="Black",fg='azure',font=('Courier',13,'bold'))
label4.place(relx=0.05,relly=0.2, relheight=0.08)

size = Entry(Frame4,font=('Century 12'))
size.place(relx=0.05,relly=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,relly=0.9, relwidth=0.25, relheight=0.05)

#Runs the window till it is closed manually
wn.mainloop()

```

5. SOURCE CODE:

```
import qrcode
from tkinter import *
from tkinter import messagebox

#Creating the window
wn = Tk()
wn.title('QR CODE GENERATOR')
wn.geometry('700x700')
wn.config(bg='Black')

#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("QR Code Generator","QR Code is saved Successfully!!")

#Label for the window
headingFrame = Frame(wn,bg="azure",bd=5)
headingFrame.place(relx=0.1,rely=0.05,relwidth=0.8,relheight=0.1)
headingLabel = Label(headingFrame, text="QR CODE GENERATOR", bg='Yellow',
    font=('Times',23,'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="Black")

Frame1.place(relx=0.1,relx=0.15,relwidth=0.7,relheight=0.3)

label1=Label(Frame1,text="Enter the text/URL:
",bg="Black",fg='azure',font=('Courier',13,'bold'))label1.place(relx=0.05,relx=0.2,
relheight=0.08)

text = Entry(Frame1,font=('Century 12'))

text.place(relx=0.05,relx=0.4, relwidth=1, relheight=0.2)

#Getting input of the location to save QR Code

Frame2 = Frame(wn,bg="Black")

Frame2.place(relx=0.1,relx=0.35,relwidth=0.7,relheight=0.3)

label2 = Label(Frame2,text="Enter the location to save the QR Code:
",bg="Black",fg='azure',font=('Courier',13,'bold'))

label2.place(relx=0.05,relx=0.2, relheight=0.08)

loc = Entry(Frame2,font=('Century 12'))

loc.place(relx=0.05,relx=0.4, relwidth=1, relheight=0.2)

#Getting input of the QR Code image name

Frame3 = Frame(wn,bg="Black")

Frame3.place(relx=0.1,relx=0.55,relwidth=0.7,relheight=0.3)

label3 = Label(Frame3,text="Enter the name of the QR Code:
",bg="Black",fg='azure',font=('Courier',13,'bold'))

label3.place(relx=0.05,relx=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="Black")
Frame4.place(relx=0.1,rely=0.75,relwidth=0.7,relheight=0.2)

label4 = Label(Frame4,text="Enter the size from 1 to 40 with 1 being 21x21:",bg="Black",fg='azure',font=('Courier',13,'bold'))
label4.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 QR',font=('Courier',13,'bold'),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()

```

6. RESULT:



QR CODE GENERATOR

QR CODE GENERATOR

Enter the text/URL:

Enter the location to save the QR Code:

Enter the name of the QR Code:

Enter the size from 1 to 40 with 1 being 21x21

GENERATE QR



QR CODE GENERATOR

QR CODE GENERATOR

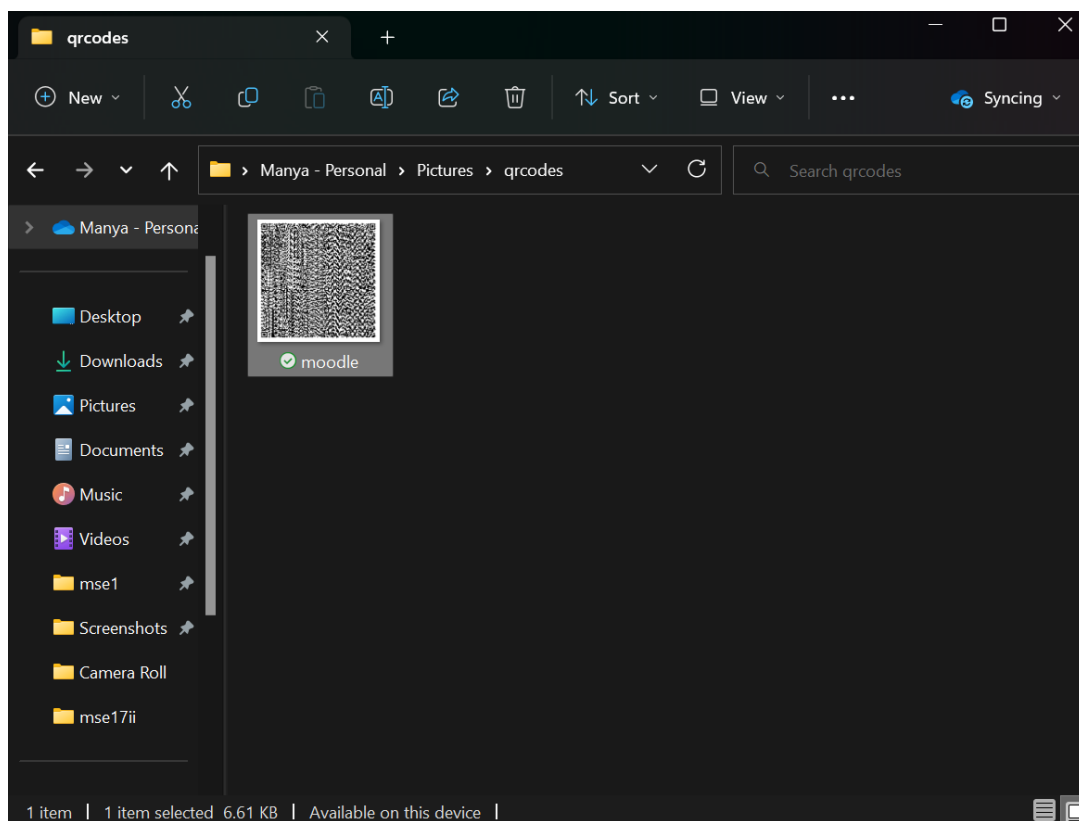
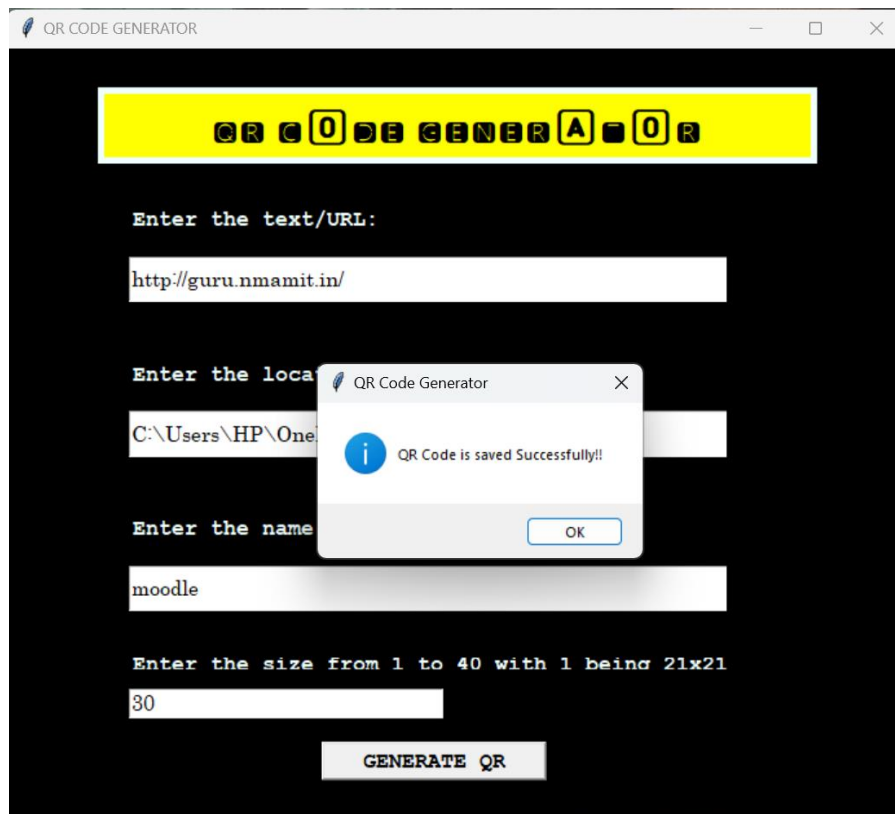
Enter the text/URL:

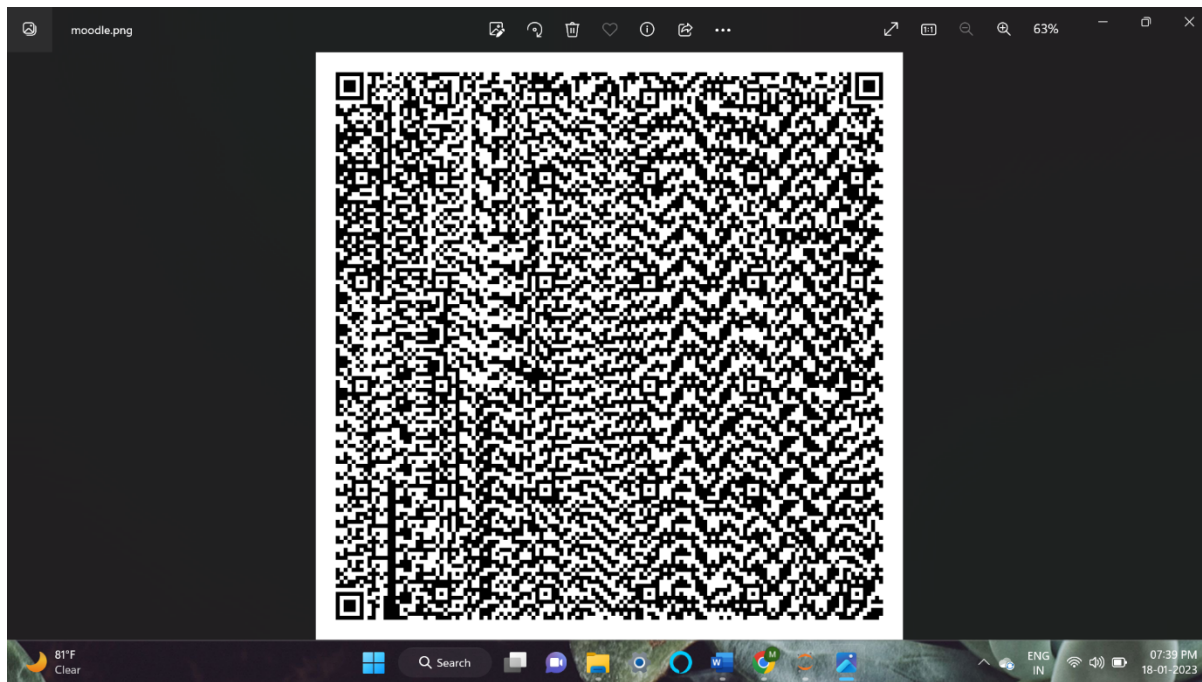
Enter the location to save the QR Code:

Enter the name of the QR Code:

Enter the size from 1 to 40 with 1 being 21x21

GENERATE QR





7. System requirements:

System requirements for Python Installation:

- Operating system: Linux- Ubuntu 16.04 to 17.10, or Windows 7 to 10.
- 2GB RAM (4GB preferable)

Pre-Requisites:

Following are the pre-requisites needed for this project:

1. Python3
2. Any IDE of your choice
3. Anaconda Distribution (Optional)

8. CONCLUSION

Among all these modules, the pyqrcode is the most useful one. The second popular is the MyQR module that leverages programmers to generate color QR or QR blended with images and logos. This makes the module more interesting. But it requires the os module to make it fully functional. So, another efficient module is the qrcode module that renders QR code with less time and space complexity.

QR code is a way of encoding more information than a traditional bar code. And most importantly, it contains information that can be easily decoded at high speed. In this paper, we

show how to create the QR codes via the web browser that facilitates users to easily create their own QR codes for websites, emails, business cards, print ads and so on. The proposed method was developed using entirely open source software such as Libqrencode, Drupal and Ubuntu. The experimental results show that the QR codes were successfully and correctly generated. Therefore, the proposed method is considerably a QR code generator collaborative tools that is available for free use.

9. References:

- Great Learning Course:
<https://www.mygreatlearning.com/academy/courses/2756984/70971>
- Codemy.com: <https://youtu.be/F5PfbC5ld-Q>
- Coding with Bintu: <https://youtu.be/UjxQRdmvC1k>
- WsCode Tech: <https://youtu.be/FOGRHBp6lvM>
- Sololearn
- Jenny's Lecturers CS IT: <https://youtu.be/DInMru2Eq6E>