

QR Scanner

By Yug Pratap Singh[25BCE11004]

yug.25bce11004@vitbhupal.ac.in

Project Report: Python QR Code Generator

Overview:-

QR codes are two-dimensional barcodes that stores some information in the form of a matrix of black and white squares. They are widely used for sharing urls and text instantly.

1. Introduction:-

This project is a desktop application developed using the Python programming language. It also provides a Graphical User Interface (GUI) that allows users to input text or links and convert them into QR codes. Unlike basic Qr generators, this is a tool that offers advanced customization and allows users to change the foreground and background colours and could also embed a custom logo into the center of the QR code.

2. Problem Statement:-

There are many online QR code generators but often they come with limitations such as:

- 1) Active internet connection.
- 2) Annoying advertisements.
- 3) Privacy concerns regarding the data input.

There is a need for a lightweight, offline and free tool that allows users to generate high-quality and customizable QR codes securely on their local machines without relying on some web server.

3. Functional Requirements:-

- 1) Data Input:** The system must accept text or URL input from the user.
- 2) Validation:** The system must warn the user if the input field is empty.
- 3) Customization:**
 - a. Allow users to pick a custom Fill Colour.
 - b. Allow users to pick a custom Background Colour.
 - c. Allow users to upload a logo .

Project Report: Python QR Code Generator

- 4) **QR Generation:** Converts the input data into a standard QR.
- 5) **Image Processing:** If a logo is uploaded program resize it and overlay it without corrupting the QR data.
- 6) **File Saving:** Open a "Save As" dialog to let the user save the final image as a PNG file.
- 7) **External Link:** Provide an "Easter egg" button that opens the history of the QR.

4. Non-functional Requirements:-

- 1) Usability:- The interface must be intuitive, fun and must require no technical knowledge to use it.
- 2) Performance:- The QR code should be generated instantly.
- 3) Reliability:- The application should handle errors without crashing.

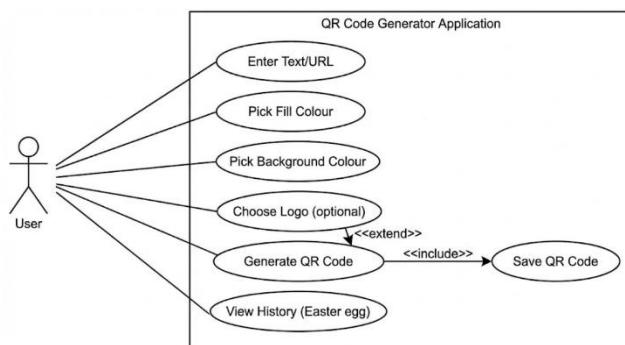
5. System Architecture:-

The application follows a simple event-driven architecture:

- 1) **Presentation Layer (Frontend):** Built using tkinter. It displays the window, buttons, and entry fields.
- 2) **Logic Layer (Backend):**
 - a) qrcode library handles the mathematical generation of it.
 - b) PIL (Pillow) handles image rendering, resizing and pasting the logo.

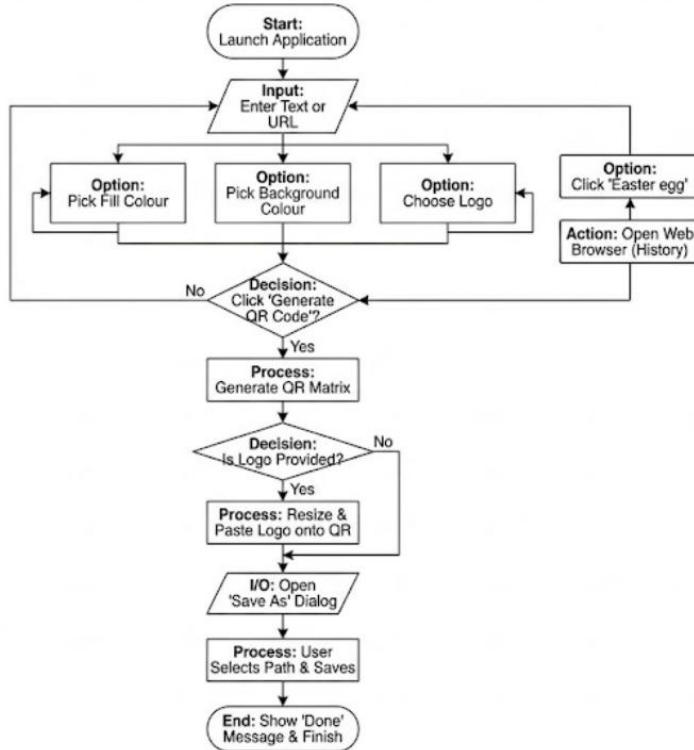
6. Design Diagrams

- 1) Use Case Diagram:-



Project Report: Python QR Code Generator

2) Workflow Diagram:-



7. Design Decisions & Rationale:-

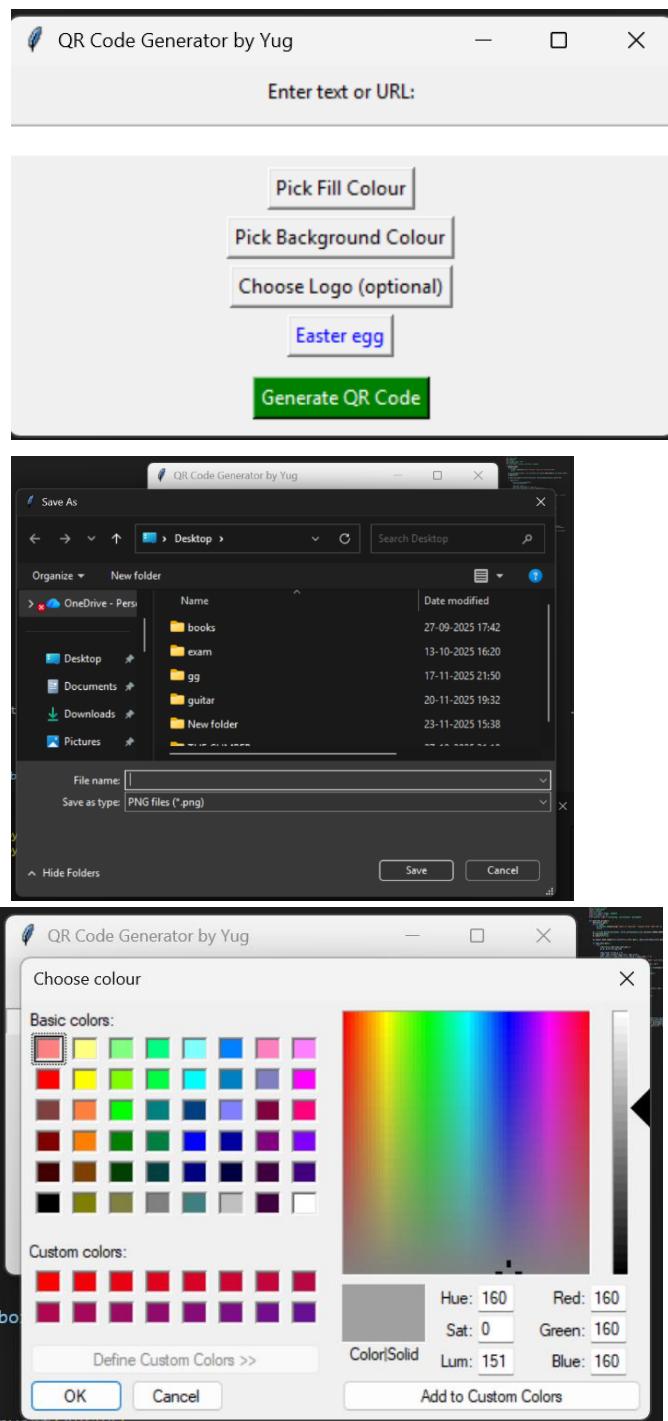
Python is used for its readability and vast library ecosystem. Tkinter is Chosen over PyQt or Kivy because it is included with standard Python installations .Pillow (PIL) is used for image manipulation because python tkinter does not support complex image operations like resizing .

8. Implementation Details

QR Code Generator uses Python 3 with the tkinter library managing the GUI elements and user interaction. The core functionality is driven by the qrcode library for matrix generation. The Pillow (PIL) library handles image manipulation, specifically resizing and pasting the optional logo onto the generated QR code with correction for scannability.

Project Report: Python QR Code Generator

9. Screenshots:-



Project Report: Python QR Code Generator

10. Testing Approach:-

The application underwent the following testing phases:

- 1) **Unit Testing**:- Verified that individual buttons (Colour Picker, File Dialog) open the correct system windows.
- 2) Functional Testing:- Verified that the "input is required" warning pops up. Tested with various image formats (.png, .jpg) to ensure the logo is centered. Scanned the generated QR codes with a smartphone to ensure the data is readable even with a logo.
- 3) Compatibility Testing:- Verified the script runs on Windows and other Python environments.

11. Challenges Faced:-

- 1) Logo Sizing:- Initially, the logo would sometimes be too large and would cover the data modules and make the QR code unscannable. I resolved this by dynamically calculating `logo_size = int(qr_w / 4)`.
- 2) Library Dependencies:- Ensuring all libraries (`qrcode`, `pillow`) were installed and linked correctly.

12. Learnings & Key Takeaways:-

Through this project I learned, How to build a functional GUI using tkinter. The logic behind QR code generation .How to manipulate images programmatically using the Pillow library. How to handle file inputs and outputs (I/O) in Python.

13. Future Enhancements:-

Future iterations will drastically improve user experience. GUI styling will transition from basic Tkinter widgets to a CSS-like framework offering a modern and visually appealing interface. Additionally, advanced image processing will be implemented using the `qrcode` library's `StyledPilImage` feature that will be providing professional-grade logo embedding with custom module shapes and optimized image resizing for enhanced scannability.

Project Report: Python QR Code Generator

14. References:-

- 1)Python 3 Documentation: docs.python.org
- 2)Tkinter Library Documentation:
docs.python.org/3/library/tkinter.html
- 3)PyPI QRCode Library: pypi.org/project/qrcode/
- 4)Pillow (PIL) Documentation: pillow.readthedocs.io