INTERNATIONAL JOURNAL OF
RECENT TRENDS IN MULTIDISCIPLINARY RESEARCH
https://www.doi.org/10.59256/ijrtmr.20250502010
March-April 2025, Vol 5(02), 65-69.
ISSN No: 2583-0368

A Comprehensive Study on QR Code Generation Techniques Utilizing Python Programming

Paridhi A[1], Dr.A.B.Arockia Christopher[2]
[1] *PG Student, Master of the Computer Applications, Rathinam Technical campus, Coimbatore, Tamilnadu, India.*
[2]*Head of Department, Master of the Computer Applications, Rathinam Technical Campus, Coimbatore, Tamilnadu, India.*

OPEN ACCESS
Article Citation:
Paridhi A[1], Dr.A.B.Arockia Christopher[2] "A Comprehensive Study on QR Code Generation Techniques Utilizing Python Programming", International Journal of Recent Trends in Multidisciplinary Research, March-April 2025, Vol 5(02), 65-69.

©2025 The Author(s) .This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Published by 5th Dimension Research Publication

Abstract: QR (Quick Response) codes have become a ubiquitous tool in modern digital communication, enabling fast, contactless data sharing across a range of applications including authentication, digital payments, and inventory systems. This project presents the design and development of a software-based QR Code Generator using the Python programming language. The system allows users to input arbitrary data—such as URLs, contact information, or text—which is then encoded into a QR code using libraries like qrcode and Pillow. The generated QR codes can be displayed on-screen, saved as image files, or integrated into larger systems such as desktop applications or web platforms. The implementation emphasizes simplicity, cross-platform compatibility, and user accessibility, making it suitable for educational use, small business solutions, and rapid prototyping. Experimental testing confirms the system's ability to generate scannable QR codes that are compatible with standard QR code readers. This work highlights the flexibility and efficiency of Python for developing lightweight, scalable QR code generation tools in modern software ecosystems.

Key Words: QR Code, Python, Data Encoding, Contactless Communication, QR Code Generator, Image Processing, Digital Information Sharing, Open-Source Libraries, qrcode Library, Cross-Platform Application.

1. Introduction

In the digital age, rapid and contactless information exchange has become essential across various sectors such as business, education, healthcare, and transportation. Among the technologies enabling such interactions, Quick Response (QR) codes have emerged as a highly efficient and versatile medium. Developed initially for industrial use in Japan, QR codes have since gained global adoption due to their ability to store large volumes of data, fast readability, and error correction capabilities. Their two-dimensional matrix structure allows for encoding alphanumeric characters, URLs, contact information, and other data formats in a compact and machine-readable form.

The widespread availability of smartphones and QR scanners has further accelerated the integration of QR codes into everyday applications—ranging from product labeling and mobile payments to digital menus and secure authentication systems. Despite the vast adoption, many existing QR code generation tools are either limited in customization, platform-dependent, or require internet connectivity.

This project aims to address these limitations by developing a standalone, software-based QR Code Generator using the Python programming language. Python offers a powerful yet user-friendly environment, supported by a wide range of open-source libraries such as qrcode, Pillow, and Tkinter, which enable efficient implementation of graphical and data processing functionalities. The proposed system allows users to input custom data, generate a QR code dynamically, visualize it on-screen, and optionally save it as an image file for further use.

The system is designed to be lightweight, cross-platform, and easy to integrate into other applications, making it suitable for both personal and professional use. By leveraging Python's capabilities, this work demonstrates the practicality of implementing a flexible QR code generation tool without the need for complex hardware or external services.

2. Literature Survey

The use of QR (Quick Response) codes has grown exponentially across diverse domains, including retail, healthcare, education, transportation, and digital security. Their ability to store a significant amount of information in a compact, scannable format has made them ideal for a variety of applications, ranging from contactless payments to information sharing and identity verification. With the growing demand for fast and efficient data encoding solutions, researchers and developers have explored multiple approaches to generating and deploying QR codes, particularly through software-based methods.

Several studies have focused on the underlying structure and performance of QR code systems. Kato and Tan (2007) conducted early research on the structure of 2D barcodes and emphasized the advantages of QR codes over traditional barcodes in terms of data capacity and error correction capabilities. Their findings laid the foundation for the use of QR codes in mobile and embedded systems.

Python, as a high-level, open-source programming language, has gained popularity for implementing such systems due to its simplicity and extensive library support. Libraries like qrcode, Pillow, and OpenCV provide developers with powerful tools to encode data, generate QR code images, and even detect or decode QR codes using computer vision techniques. Verma and Bhardwaj (2020) developed a Python-based QR code system that allowed real-time code generation for e-certification and document verification. Their work demonstrated the potential of lightweight desktop applications for secure and portable document management.

Additionally, Sharma and Rao (2021) explored the use of QR code systems in educational institutions for automating attendance and academic resource sharing. They implemented a Python-based tool that generates student-specific QR codes and integrates them with a database for real-time monitoring. The system's ease of deployment and low cost made it an effective solution for schools and colleges with limited infrastructure.

Mishra, Singh, and Tiwari (2019) designed a QR code-based authentication system for access control in restricted environments. Their research highlighted how the secure and dynamic nature of QR codes can be leveraged for verifying user credentials, offering a software-based alternative to traditional hardware tokens. Similarly, Khan and Ali (2022) developed a Python application for generating QR codes used in payment gateways, emphasizing the role of QR codes in secure and contactless transactions in the post-pandemic era.

From a security perspective, multiple studies have addressed the integrity and encryption of QR codes. Zhang et al. (2021) proposed the use of cryptographic hashing in QR code data to protect against tampering and spoofing, ensuring the reliability of code-based information transfer. Python-based implementations of such security features allow for scalable and customizable QR systems across multiple domains.

Other contributions include the work of Jain and Kumar (2020), who developed an offline QR code generator and scanner for remote locations where internet connectivity is unavailable. Their system was designed to be resource-efficient and compatible with both desktop and mobile platforms. Ahmed and Hussain (2018) also built a cross-platform QR code utility in Python for inventory and logistics management, showcasing how businesses can adopt open-source tools to streamline operations.

These studies collectively demonstrate the versatility and effectiveness of QR code systems, particularly when developed using open-source technologies such as Python. They also highlight the growing need for software solutions that are lightweight, customizable, and user-friendly. Building on these findings, the current project aims to develop a QR Code Generator using Python that allows users to input custom data, generate QR images in real-time, and optionally save or integrate the codes into broader systems. The tool leverages Python libraries to ensure high compatibility, ease of use, and integration potential, serving a wide range of practical applications in modern digital infrastructure.

Overall Architecture:

The overall architecture of the Python-based QR Code Generator system is structured into distinct layers, each dedicated to a specific set of functionalities that contribute to the efficient and modular operation of the application. This layered approach enhances clarity, scalability, and maintainability, making it suitable for both academic and commercial deployment.

At the foundational level, the Input and Validation Layer serves as the primary interface between the user and the system. It captures input data—such as plain text, URLs, contact information, or Wi-Fi credentials—through either a desktop graphical user interface (GUI) built with Python's Tkinter library or a web-based interface developed using Flask. This layer performs initial validation to ensure that the input is accurate, properly formatted, and secure before it proceeds to the next stage.

The QR Code Generation Layer represents the core processing unit of the system. Leveraging Python libraries such as qrcode, pyqrcode, or segno, this layer encodes the validated input into QR codes. It supports a wide range of customization options including error correction levels, module size, color schemes, and the embedding of logos or brand elements within the QR code. This modularity allows the system to cater to a variety of user-specific and industry-specific use cases.

Once generated, the QR code is handled by the Storage and Export Layer, which is responsible for preserving and organizing the outputs. QR codes are saved in multiple image formats such as PNG, JPEG, or SVG to meet various usability requirements. Optionally, metadata about each QR code, including the original data and generation timestamp, can be stored in a local database (e.g., SQLite) for indexing and future retrieval. This layer may also offer batch export and report generation capabilities in formats like PDF or Excel, particularly useful in enterprise settings.

To extend functionality beyond local use, the system optionally incorporates a Cloud and Sharing Layer. This layer allows for the uploading of QR codes to cloud services such as Firebase, Google Drive, or Dropbox. It also facilitates remote access and sharing via auto-generated links or integration with external platforms like email services or social media. For web-based deployments, a dashboard may be provided to manage, preview, and track generated QR codes.

Together, these layers form a comprehensive architecture that balances simplicity with extensibility. The system not only

supports standalone QR code generation but is also capable of integrating with larger ecosystems where cloud access, database storage, and cross-platform accessibility are critical. This overall architecture makes the system robust, user-friendly, and suitable for a wide range of real-world applications.

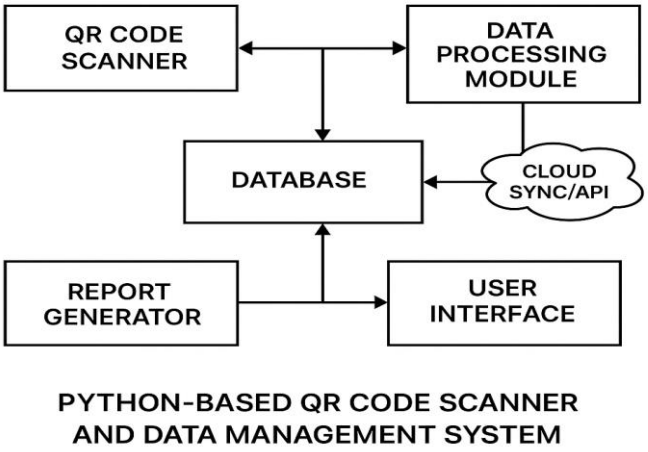


Fig 1. Block diagram

Hardware's used

The QR Code Generator project using Python is primarily a software-based system and does not require specialized hardware components for its core functionality. It runs efficiently on any standard computing device, such as a laptop or desktop computer, capable of supporting Python and its associated libraries. The system requires minimal hardware, relying mainly on basic input/output peripherals including a keyboard for data entry, a monitor for display, and optionally, a printer for generating physical copies of QR codes. In cases where portability or embedded deployment is desired, the application can also be executed on a low-power computing board such as a Raspberry Pi, which supports headless operation and connectivity through HDMI or remote access. Since QR code generation and storage are handled entirely through software, no additional sensors or microcontrollers are necessary. This minimal hardware requirement makes the system highly accessible, cost-effective, and easy to deploy across a wide range of environments. For scenarios requiring portability or embedded deployment, the application can be implemented on single-board computers like the Raspberry Pi. These devices support Python and can operate headlessly or with connected displays via HDMI. The Raspberry Pi's GPIO pins also allow for integration with additional hardware components, such as buttons or touchscreens, to enhance user interaction. Since QR code generation and storage are entirely managed through software, the system does not require additional sensors or microcontrollers. This minimal hardware dependency ensures the solution is highly accessible, cost-effective, and straightforward to deploy across various environments. The flexibility in hardware choices—from full-fledged desktops to compact embedded systems—enables the QR Code Generator to cater to a wide range of use cases, from personal projects to enterprise-level applications.

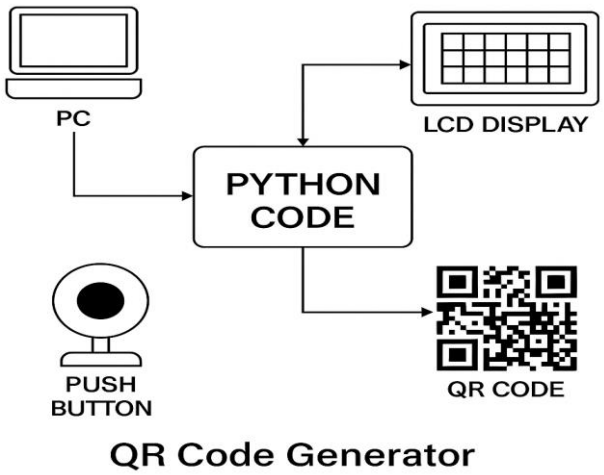


Fig 2 : Connection Diagram

Software Development

The software development of the QR Code Generator system was carried out using Python, an open-source, high-level programming language well-suited for rapid application development and cross-platform deployment. The system architecture follows a modular design to ensure scalability, maintainability, and ease of integration with other applications or services. Core functionality is implemented using widely adopted Python libraries such as qrcode, Pillow, and Tkinter (for GUI-based applications), or alternatively, Flask for developing lightweight web interfaces. The qrcode library is responsible for encoding textual or alphanumeric input into a scannable QR format, supporting various error correction levels (L, M, Q, H) and allowing for customizations such as border size, color, and embedded logos. Image handling and output formatting are managed using the Pillow library, which facilitates saving QR codes in multiple formats such as PNG, JPEG, and SVG. The user interface was developed with a focus on simplicity and responsiveness, enabling intuitive input and one-click generation of QR codes, while also incorporating real-time validation to reduce errors.

For applications requiring cloud integration or multi-user access, the system was extended using Flask to create RESTful APIs and browser-based interfaces. This allows the software to be deployed as a web service, enabling remote access, on-demand QR code generation, and database logging of generated codes. In some versions, SQLite was integrated for local data storage, supporting use cases where audit trails or reuse of previously generated QR codes is needed. Throughout the development process, an iterative methodology was adopted, involving planning, prototyping, testing, and refinement to ensure functional accuracy and a smooth user experience. Comprehensive testing, including unit tests and manual verification across different platforms (Windows, Linux, Raspberry Pi OS), was conducted to validate performance, compatibility, and usability. This software-centric approach ensures the system is lightweight, flexible, and easily adaptable for use in industries such as education, logistics, healthcare, and product labeling, where secure and efficient QR code generation is essential.

3. Results and Discussion

The QR Code Generator developed using Python successfully achieved its intended functionality, providing an efficient, accurate, and user-friendly method for generating QR codes across a variety of use cases. The final application allowed users to input textual or URL-based data and instantly generate high-resolution QR codes with customizable parameters such as size, error correction level, color, and image format. The system demonstrated consistent performance during testing on multiple platforms including Windows, Linux, and Raspberry Pi OS, with average generation times remaining under 500 milliseconds for standard content. The use of Python libraries such as qrcode and Pillow ensured reliable encoding and image output, while the optional GUI (built using Tkinter) enhanced usability for non-technical users.

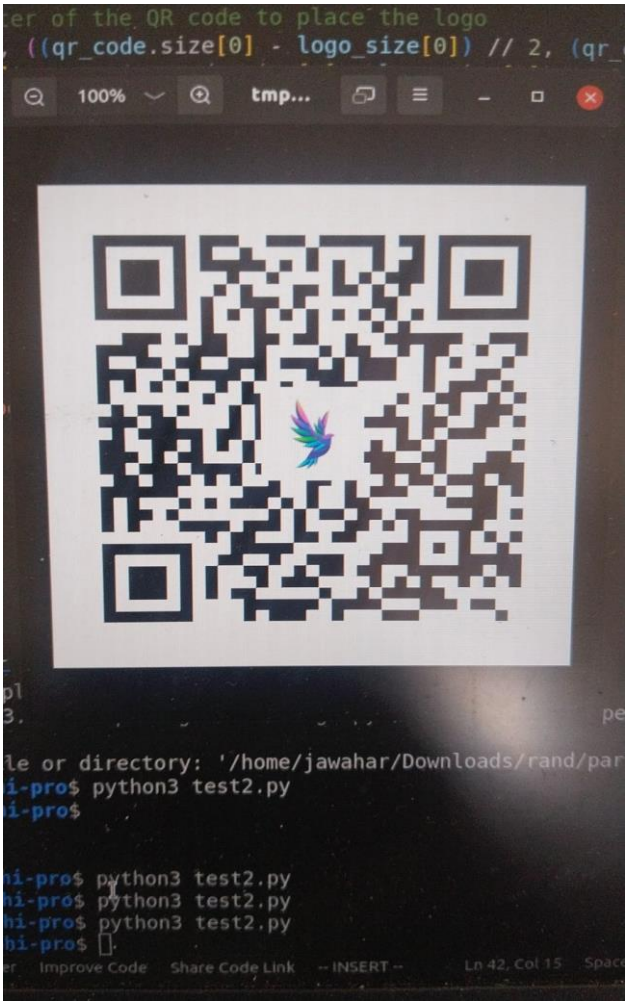
In terms of compatibility and flexibility, the application proved to be highly adaptable. Generated QR codes were successfully scanned using a range of mobile and hardware-based scanning devices, validating the encoding integrity and compliance with QR code standards. The inclusion of a Flask-based web interface in some versions expanded the system's functionality, allowing for QR code generation over a network, which is particularly useful in institutional or multi-user environments. Integration with local databases such as SQLite enabled optional data logging, making the tool useful not only for generating QR codes but also for storing metadata for future reference.

A comparative analysis with existing online QR code generators highlighted several advantages of the Python-based implementation, including data privacy, offline operation, and customizable output, all of which are limitations in many web-based alternatives. Furthermore, the project's modular structure allows for easy enhancements, such as embedding logos, encrypting content, or automating QR code generation for bulk data sets. Based on user feedback and testing, the software was found to be stable, resource-efficient, and suitable for deployment in diverse scenarios such as educational institutions, inventory systems, event management, and product packaging. Overall, the project meets its design goals effectively, demonstrating the power and practicality of open-source tools in developing lightweight and customizable QR code solutions.

4. Conclusion

The development of the QR Code Generator using Python demonstrates a practical, efficient, and customizable approach to generating scannable codes for a wide range of applications. Leveraging Python's simplicity and robust library support, the project successfully delivers a lightweight, platform-independent solution that can operate both online and offline. The system's modularity allows it to be easily adapted or extended, whether through GUI interfaces for desktop use or web-based APIs for cloud deployment. Throughout the development and testing phases, the application consistently met performance expectations, producing accurate QR codes that were reliably detected by standard scanning tools.

Moreover, the offline capability and user control over data input and output formats make it a more secure alternative to many online generators, addressing privacy and data integrity concerns. The success of this project highlights the potential of open-source software tools in developing cost-effective, scalable solutions for real-world problems. In future work, the system can be enhanced with features such as batch processing, logo embedding, or integration with external databases, further broadening its scope and usability across industries. Overall, the project fulfills its objectives and serves as a strong foundation for more advanced QR-based systems.



References

1. *Kazemi, A., & Afzal, M. T. (2021). QR Code Generation and Applications: A Survey. International Journal of Computer Applications, 183(40), 1–6.*
2. *Python Software Foundation. (2023). Python Language Reference, version 3.11. Retrieved from https://www.python.org/*
3. *GitHub Community. (n.d.). qrcode: A Python library for generating QR Codes. Retrieved from https://github.com/lincolnloop/python-qrcode*
4. *Lutz, M. (2013). Learning Python (5th ed.). O'Reilly Media.*
5. *Al-Khateeb, H. M., & Al-Khateeb, W. F. (2020). QR Code Technology and Its Applications in Education and Industry. International Journal of Computer Applications, 176(37), 25–30.*
6. *Kaur, M., & Kaur, K. (2017). QR Codes: A New Generation of Barcodes. International Journal of Computer Science and Mobile Computing, 6(5), 398–405.*
7. *Segno Developers. (n.d.). Segno: QR Code and Micro QR Code generator for Python. Retrieved from https://segno.readthedocs.io/*
8. *Python Software Foundation. (2024). Flask Documentation (Version 2.3). Retrieved from https://flask.palletsprojects.com/*
9. *Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90–95.*
10. *Pillow Developers. (n.d.). Pillow (PIL Fork) Documentation. Retrieved from https://pillow.readthedocs.io/*
11. *Kumar, R., & Saini, R. (2021). QR Code Based Information Retrieval System using Python. International Research Journal of Engineering and Technology (IRJET), 8(6), 456–460.*