Bulletin of the *Transilvania* University of Braşov • Vol. 18(67) No. 2 – 2025
Series I: Engineering Sciences
https://doi.org/10.31926/but.ens.2025.18.67.2.1

QR CODE GENERATOR WEB APPLICATION WITH FREE CUSTOMIZATION AND ANALYTICS FEATURES

O. PAVLOV[1] **D. E. ILEA GHIȚĂ**[1]

Abstract: *The QR Code Generator proposed in this paper is a web-based platform designed to facilitate the creation, customization, and analytics of QR codes for diverse applications, including marketing and communication. All functionalities are open source and are fully accessible through the website. This work is motivated by the scarcity of freely available online services offering comprehensive functionality and advanced customization, as most existing solutions are restricted to limited trial periods. This paper presents the algorithm underpinning the QR code generation process. In addition, the entire application is open source and may be freely deployed or hosted by third parties.*

Key words: *QR Code Generation, QR Code Customization, Analytics, Algorithm Design*

1. Introduction

Quick Response (QR) codes have become a widely adopted technology for bridging physical and digital interactions. Originally developed for industrial use, they are now employed across diverse sectors such as marketing, retail, healthcare, logistics, and education. Their abilities to store a large amount of data, ease of scanning with mobile devices, and flexibility in linking to websites, applications, or multimedia content make them an efficient tool for information dissemination and customer engagement.

Despite their popularity, many online QR code generation services remain limited. Free versions often restrict essential features such as customization, analytics, or bulk generation, while advanced functionality is typically locked behind paid subscriptions or time-limited trials. This creates a barrier for small organizations, researchers, educators, and individual users who require robust and flexible QR code tools without incurring ongoing costs.

An openly accessible, web-based QR code generation platform addresses this gap. By offering unrestricted creation, customization, and analytics capabilities at no cost, such a platform democratizes access to QR code technology and empowers users to integrate it

[1] Faculty of Electrical Engineering and Computer Science, *Transilvania* University of Braşov.

into their activities more effectively. Furthermore, making the platform open source allows for transparency, community-driven improvements, and independent hosting, ensuring that users are not dependent on a single provider for long-term access. The proposed solution is characterized by a well-organized implementation that facilitates analysis and maintenance, thereby offering an advantage over the majority of open-source alternatives.

A QR code is a two-dimensional barcode originally developed in 1994 by the Japanese company Denso Wave [7]. QR codes conform to a standardized encoding format that must be strictly adhered to in order to ensure reliable scanning and interpretation by devices worldwide.

The encoding and decoding processes underlying QR codes employ well-established mathematical frameworks, including the Reed–Solomon error-correction algorithm, the Bose–Chaudhuri–Hocquenghem (BCH) algorithm, and operations over Galois fields. In these algorithms, polynomials defined over Galois fields are used to encode and decode the stored data.

QR codes support multiple data modes, each specifying the set of symbols that can be encoded. Four primary modes are defined: Numeric (digits), Alphanumeric (digits, Latin characters, and selected symbols), Binary (ISO/IEC 8859-1, an ASCII-based character-encoding standard), and Kanji (Shift JIS X 0208, a character set for Japanese language). These modes allow QR codes to store plain text and other data types.

Additionally, QR codes provide four levels of error correction, which determine the proportion of data that can be recovered after loss or damage. Depending on the selected level, between 7 % and 30 % of the original data can be restored. This variation arises from the differing amounts of error-correction code words added to the data: higher correction levels improve recoverability at the cost of reduced effective data capacity.

The main contribution of this paper consists in detailing the implementation of a flexible open source package with free customization and analytic features. The construction of the algorithm is implemented in TypeScript programming language in order to be easily embedded into web resources.

2. State of the Art Survey

The QR code generation algorithm is standardized, and numerous applications currently implement it. As a result, most current research on QR codes focuses on enhancing specific functionalities or developing novel applications based on this technology. One prominent area of study is security, aimed at ensuring privacy and safeguarding the transmission of sensitive information. Approaches to enhancing security include sequential masking techniques [3] as well as encryption schemes employing public and private key algorithms [1].

Another active research direction involves customization and visual design to increase the applicability of QR codes in marketing and business contexts. Common techniques include embedding brand logos within the QR code or styling codes with brand-specific colors, which have become widespread in modern marketing practices. Recent studies have also investigated the effects of placing images over QR codes and embedding QR codes within larger images, assessing their impact on both scan reliability and aesthetic integration [4].

The transition from static to dynamic QR codes has significantly enhanced their capabilities, particularly in data tracking and analytics. Dynamic QR codes utilize short URLs that can be redirected to different destinations without altering the physical code, allowing for real-time updates, content management and activity tracking. This flexibility facilitates personalized customer interactions and targeted marketing strategies.

Recent studies indicate a substantial increase in the adoption of dynamic QR codes. For instance, 79% of businesses now employ dynamic QR codes for context-aware interactions, and global scan volume has surged by 433% since 2021 [5], [11]. Furthermore, integrating dynamic QR codes with analytics platforms like Google Analytics 4 enables detailed tracking of user interactions, including scan locations, devices used, and engagement metrics [6].

Advancements in diffusion-based models have led to the development of QR code generators like DiffQRCoder, which employ Scanning-Robust Perceptual Guidance (SRPG) to maintain scanning robustness while enhancing visual aesthetics. This approach ensures that QR codes remain scannable even under varying conditions, such as different scanning angles and lighting environments [2].

3. Proposed Methodology

3.1. The algorithm of QR code generation

A QR code encodes text data and is characterized by chosen error-correction level. Four standardized error-correction levels are defined: L (approximately 7 % of codewords can be restored), M (15 %), Q (25 %), and H (30 %). These percentages represent the maximum proportion of data that can be recovered after random damage or data loss. Higher levels of protection require the inclusion of more error-correction blocks, thereby reducing the effective data capacity. Selection of the error-correction level is typically guided by the intended use of the QR code. In QR-code generation, the data to be encoded together with the chosen error-correction level constitute the input to the encoding algorithm, while the output is the QR-code matrix.

Because a QR code is a square matrix, its size (dimension) must also be specified. In the QR-code standard, this dimension is determined by the version number, which ranges from 1 to 40. The formula that connects the version number and the dimension is the following

$$Dimension = Version * 4 + 17$$

The first version of the QR code standard specifies a matrix of 21 × 21 and each subsequent version increases the dimension of the matrix by four on each side.

During encoding, the version number is selected by the algorithm to be the smallest one that can accommodate the input message together with the required error-correction blocks for the chosen level. Once a candidate version is determined, the encoder typically attempts to use the highest possible error-correction level such that both the message and the corresponding error-correction blocks fit within the matrix. For example, if a client selects error-correction level M and the minimal version X can accommodate the message and its error-correction codewords but the message at level Q can also be accommodated, then level Q is selected to provide maximum protection.

After determining the version and error-correction level, the algorithm proceeds to encode the input data. Because a QR code consists of a square matrix of black and white components, the encoded data can be represented as a binary array, with white modules corresponding to a logical 0 and black modules to a logical 1.

The encoding process starts with the construction of a raw bit stream. This stream consists of several components: a 4-bit mode indicator specifying the encoding mode; the message length expressed in binary format; the message characters in consecutive byte form; and a 4 zero bits terminator. If the QR code capacity exceeds the number of bits in the constructed bitstream, padding bytes (0xEC and 0x11) are appended until the required length of bits is reached. It is allowed to add only several bits of the padding byte to the end to have the exact needed number of bits in the stream. These particular padding values are selected because their bit patterns alternate, which improves scanner reliability, and they do not alter the transmitted message content.

Then, using the specification, the raw bitstream is split into “groups” and “blocks”. The idea is to spread the risk of damage to the whole QR code. It would potentially allow to restore the data using error correction bits. After the split, each block is processed with Reed-Solomon algorithm to create error correction bits. These bits help restoring data after the damage. At this step, the modular arithmetic and polynomial properties are applied. From this step, each block has its error correction codewords.

After the error correction codewords are obtained, the blocks with their error correction codewords can be interleaved – connected back to a stream. Keeping the blocks order, the algorithm connects blocks and their codewords back into the stream. Codewords become part of a block and are interleaved together with the blocks.

At this stage, the final bitstream is generated and can be placed into the QR code matrix. Before inserting the data bits, the algorithm reserves specific areas of the matrix for functional patterns. The set of required patterns varies depending on the QR code version.

After reserving the required space of the matrix, the data bits are placed in the remaining cells following a “zig-zag” pattern starting at the bottom-right corner. Subsequently, the metadata for the QR code is inserted into designated positions within the matrix. This metadata includes the error-correction level and the selected mask pattern. The mask is applied to increase the alternation of light and dark modules, thereby improving scan reliability. There are eight mask patterns, each represented by a Boolean function that determines which modules must invert their colour. These functions take as input the x and y coordinates of the cell, counted from the top-left corner beginning at (0, 0). The algorithm evaluates all mask patterns and evaluates the one that maximizes bits alternation.

The metadata and the masked bitstream are then inserted into the matrix, completing the QR code. It is recommended to add a quiet zone—a white border surrounding the matrix—with a width equivalent to three to four modules.

3.2. QR Code Generator application in Python

The features of the proposed application are accessed via the website and the server is written in Python. The application is built using the Flask framework that provides methods and classes to build REST API applications.

Backend web applications that implement REST API have a set of defined methods (endpoints) that the client (frontend application – a website in browser) can use. Each call or request has a specific structure – it has an address or endpoint that specifies the function that has to be called, set of headers that specify the metadata about the request and body that is optional and can contain the data for the request. The body can be associated with the arguments of the function.

The data used in the proposed application is stored in MySQL Database. Flask SQL Alchemy library is used for SQL queries. The app is deployed via PythonAnywhere free hosting [10].

The application supports and requires authentication for the users to provide them private statistics and analytics. The authentication is made by a login-password pair. Passwords are stored in database in hashed version so the access to the user's account is protected.

The source code of this backend application can be accessed with this link: https://github.com/Alekseypavlov14/competition-qr-codes-backend.

3.3. QR Code Generator frontend application

The frontend application is a website that can be opened and used using browsers like Google Chrome, Mozilla Firefox or others. This application uses the backend application. The data is passed between two applications using a HTTP protocol which is standard for web applications.

This frontend application is written using TypeScript programming language. It is a modern language used in application development and is compiled in JavaScript code. The codebase is compiled in HTML/CSS/JS code files that are native for the web browsers.

The application has a homepage with the list of all the created QR codes. Each QR code corresponds to the page with the data about that code. There is a form to customize the QR code colours and design. The variants can be separately downloaded in different formats (SVG, PNG, JPEG, WEBP) and they all lead to the specified data. These variants can be used independently in different resources using the analytics feature.

The website is accessible with this link: https://qr-coder-io.netlify.app.

The source code of the frontend module is accessible with this link: https://github.com/Alekseypavlov14/competition-qr-codes-frontend.

3.4. Standalone open-source package for QR code generation

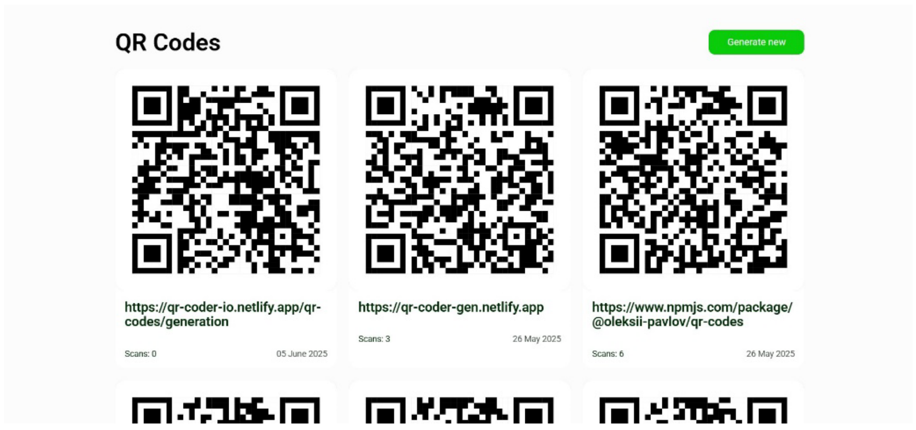
The package that includes the functions to generate, display and download QR codes in different formats is written in TypeScript programming language. It is accessible with NPM [9] at the link (https://www.npmjs.com/package/@oleksii-pavlov/qr-codes) and is open source. The usage and modification of this package is free.

4. Results and Discussion

The application is ready to use and it is possible to create more clients for this QR code generator using a standalone free open-source package. The usage is free, highly customizable and includes all the features that are needed for QR code generation for different purposes.

The open-sourced modules can be hosted with no costs involved. They can also be modified if needed. The source code of the system modules, including the core generation module, is structured in accordance with established best practices and design patterns. Owing to the high quality and maintainability of the code, the solution is readily extensible and amenable to further development. The system exhibits a negligible response time, as the algorithm executes without perceptible time delays. The algorithm of QR code generation that is provided by the standalone module requires 50 ms averagely per one code from a message of 50 characters. These characteristics constitute significant advantages of the proposed implementation. Furthermore, the service is free of advertisements, thereby contributing to a satisfactory user experience. Customization and analytical features are freely accessible and are easy to use.

This application is already used by local department of non-commercial organization Terre des Hommes when it is needed to generate QR Codes without limits for the expiring time. In Figure 1, it is displayed the homepage of the web platform. The user can access previously generated QR codes. In Figure 2, the generation process form is presented. The generation consists of specifying the message and minimal error-correction level and choosing colours and design pattern. In Figure 3, the customization panel is presented to provide a feature to customize QR code after it is generated with the form displayed in Figure 2.

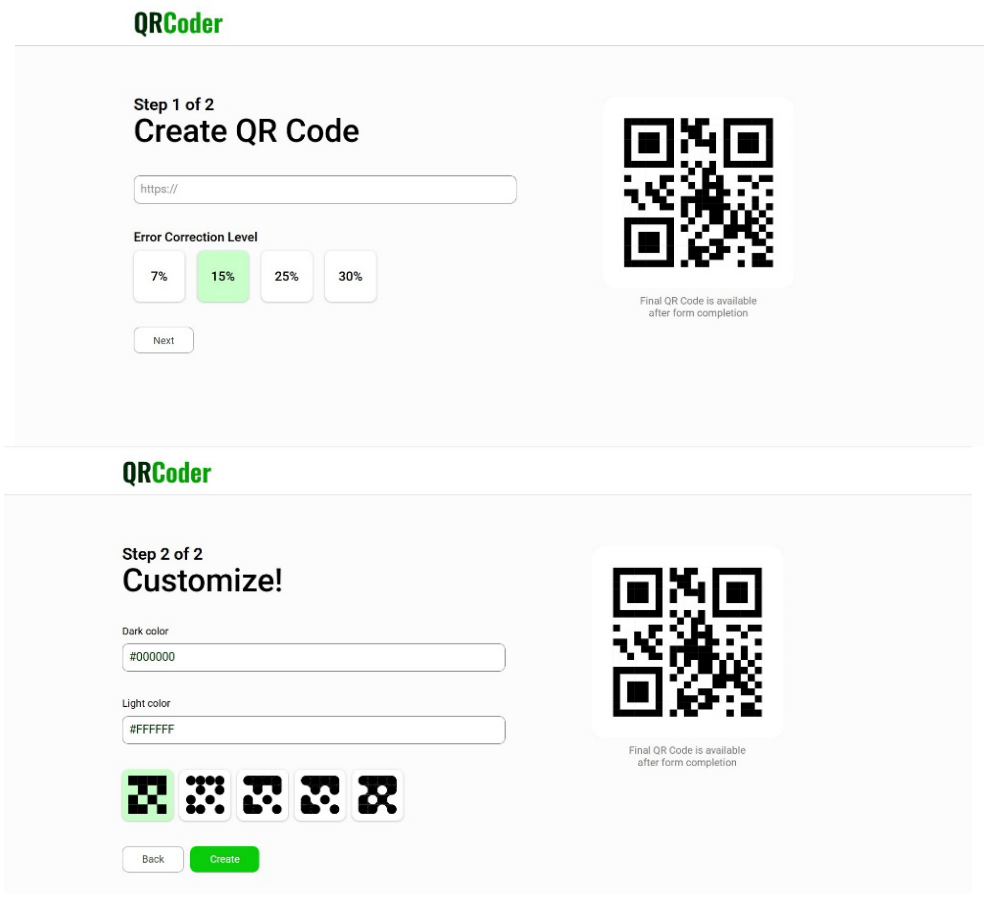


https://qr-coder-io.netlify.app/qr-codes/generation
Scans: 0 05 June 2025

https://qr-coder-gen.netlify.app
Scans: 3 26 May 2025

https://www.npmjs.com/package/@oleksii-pavlov/qr-codes
Scans: 6 26 May 2025

Fig. 1. *Website homepage*



QRCoder

Step 1 of 2

Create QR Code

https://

Error Correction Level

7% [✓ 15%] 25% 30%

Next

Final QR Code is available after form completion

QRCoder

Step 2 of 2

Customize!

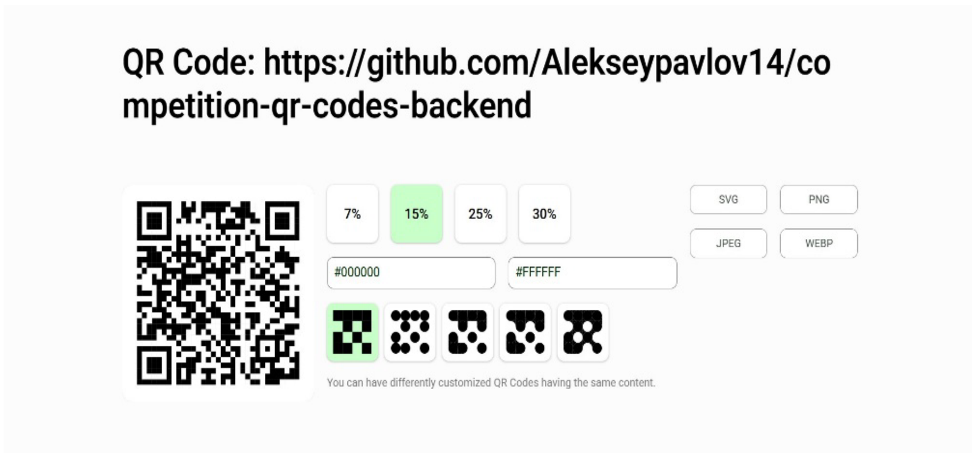
Dark color: #000000

Light color: #FFFFFF

Back Create

Final QR Code is available after form completion

Fig. 2. *QR Code Generation form in 2 steps*



QR Code: https://github.com/Alekseypavlov14/competition-qr-codes-backend

7% [✓ 15%] 25% 30%

#000000 #FFFFFF

SVG PNG JPEG WEBP

You can have differently customized QR Codes having the same content.

Fig. 3. *QR Code Customisation form*

5. Future Work

The functionality of the proposed system can be further extended in several ways. One promising enhancement, although not directly related to the QR code generation algorithm itself, is the integration of branding capabilities—specifically, the ability to embed a company or product logo at the center of a QR code without compromising its scannability. In addition, support for a broader range of character sets could be implemented, enabling the encoding of diverse alphabets and thereby increasing both the amount and the linguistic diversity of the information stored within a single QR code.

Additionally, the application backend module could be enhanced through the embedding to Docker containers [8], further simplifying the deployment process. Although the current deployment procedure is already straightforward, containerization would provide additional portability and consistency across different environments.

References

1. Ahamed, S., Mustafa, H.: *A Secure QR Code System for Sharing Personal Confidential Information.* In: Proceedings of the 2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2), Rajshahi, Bangladesh, July 2019, p. 2-3.
2. DiffQRCoder: *Diffusion-based Aesthetic QR Code Generation with Scanning Robustness Guided Iterative Refinement*, arXiv:2409.06355
3. Pasala, A., Mukherjee, S.: *Variable masking pattern-based QR codes for high security.* In: Proceedings of the 2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, July 2024, p. 1-2.
4. Nandhini, S.: *Performance evaluation of embedded color QR codes on logos.* In: Proceedings of the 2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM), Chennai, India, March 2017, p. 2-4.
5. https://qr-kode.no/fr/blog/how-to-create-dynamic-qr-codes-with-analytics. Addressed: 27.09.2025
6. https://www.analyticsmania.com/post/track-qr-codes-with-google-analytics-4. Addressed: 27.09.2025
7. https://www.denso-wave.com/en. Addressed: 26.09.2025
8. https://www.docker.com. Addressed: 26.05.2025
9. https://www.npmjs.com. Addressed: 26.09.2025
10. https://www.pythonanywhere.com. Addressed: 25.09.2025
11. https://www.uniqode.com/blog/qr-code-insights/qr-code-report. Addressed: 27.09.2025