



**SIMATS**  
ENGINEERING



**SIMATS**  
Saveetha Institute of Medical And Technical Sciences  
(Declared as Deemed to be University under Section 3 of UGC Act 1956)

## **CAPSTONE PROJECT REPORT**

### **PROJECT TITLE**

QR Code GENERATION USING C++

### **TEAM MEMBERS**

192211920 K Prasanth

192224118 Shaik Muzamil

### **COURSE CODE / NAME**

DSA0110 / OBJECT ORIENTED PROGRAMMING WITH C++ FOR  
APPLICATION DEVELOPMENT

SLOT A

### **DATE OF SUBMISSION**

12.11.2024



**SIMATS**  
ENGINEERING



**SIMATS**  
Saveetha Institute of Medical And Technical Sciences  
(Declared as Deemed to be University under Section 3 of UGC Act 1956)

## **BONAFIDE CERTIFICATE**

Certified that this project report \_QR GENERATION C++\_\_\_\_\_

is the Bonafide work of K Prasanth (192211920)

SK Muzamil(192224118)\_\_\_\_\_

\_\_\_\_\_who

carried out the project work under my supervision.

SUPERVISOR

**DR S SANKAR**

## ABSTRACT

This project presents a system for web page generation using QR codes, implemented in C++. The system aims to provide an efficient and user-friendly method for creating and accessing web content through QR codes, bridging the gap between physical and digital worlds. By encoding URLs and web content into QR codes, users can seamlessly transition from printed materials to online resources. The proposed system demonstrates the practicality of using QR codes for web content generation and access. It is particularly useful in contexts where quick and direct access to digital information is required, such as in educational settings, marketing campaigns, and event management. The C++ implementation ensures that the system is both robust and performant, capable of handling a variety of use cases with ease. The project concludes with a discussion on potential enhancements, such as incorporating error correction mechanisms, supporting additional data formats, and extending the system to mobile platforms. Through this project, the integration of QR code technology with web content generation is explored, showcasing its potential to streamline the user experience in accessing digital information

**Keywords:** Web Generation, QR-code, Object-oriented Programming, Code scanner, websites, C++ libraries

# INTRODUCTION

## 1.1 INTRODUCTION

In today's digital age, the need for seamless integration between physical and digital information has become increasingly important. Quick Response (QR) codes have emerged as a powerful tool for bridging this gap, offering a fast and efficient way to link printed materials to online resources. QR codes are two-dimensional barcodes that can store a variety of data, including URLs, text, and other forms of information, which can be easily accessed using a smartphone or any QR code scanner. This project focuses on the development of a system for web page generation using QR codes, implemented in C++. The primary goal is to enable users to generate and access web content through the simple scan of a QR code, enhancing the accessibility and distribution of information. By leveraging the capabilities of C++, the system ensures high performance, efficient memory usage, and the flexibility needed to handle diverse types of web content. The project concludes with a discussion on potential enhancements, such as incorporating error correction mechanisms, supporting additional data formats, and extending the system to mobile platforms. Through this project, the integration of QR code technology with web content generation is explored, showcasing its potential to streamline the user experience in accessing digital information.

the intricacies of this project, we embark on a journey to unlock the potential of computer vision in revolutionizing traffic management. From the pixels captured by roadside cameras to the algorithms that analyze and interpret them, our quest is to harness the power of technology to create safer, more efficient urban environments. Real-Time Web generation by qr code refers to the application of computer vision techniques, such as deep learning algorithms and image processing, to detect and interpret traffic lights in real-time from live camera feeds. This technology aims to improve road safety and traffic flow by accurately identifying traffic signals and facilitating timely decision-making in transportation systems.



Fig 1: QR Generator

## 1.2 SCOPE OF THE STUDY

Studying web generation for QR codes using C++ encompasses a wide scope that blends theoretical understanding with practical application. Here are key aspects to consider:

1. **Fundamental Concepts:** Begin by exploring the principles behind QR code generation, such as data encoding methods (numeric, alphanumeric, binary), error correction levels (L, M, Q, H), and QR code structure (alignment patterns, timing patterns, version control).
2. **Algorithm Development:** Dive into the algorithms used for QR code generation, such as Reed-Solomon error correction, byte encoding, and masking patterns. Understanding these algorithms is essential for implementing efficient and accurate QR code generation in C++.
3. **Library Utilization:** Familiarize yourself with existing C++ libraries for QR code generation, such as `qrcodegen` or `libqrencode`, and learn how to integrate them into web applications. This involves understanding library APIs, customization options, and performance considerations.
4. **Web Integration:** Explore methods to integrate QR code generation into web applications using C++. This includes generating QR codes dynamically based on user input, embedding QR codes in HTML/CSS for responsive design, and leveraging JavaScript for enhanced interactivity.
5. **Cross-Platform Compatibility:** Consider the challenges and solutions for ensuring QR code generation works seamlessly across different platforms (desktop, mobile) and browsers. This may involve testing and optimizing C++ code for performance and compatibility.

## LITERATURE REVIEW

WEB GENERATION OF QR CODE USING OBJECT ORIENTED PROGRAMMING

AUTHOR: ZHENJI, WHANG LI, J., & MENJHILO, J YEAR: 2019 2.1

OVERVIEW Algorithm development plays a pivotal role in this study, as it involves implementing complex mathematical algorithms within C++ to generate QR codes accurately.

QR (Quick Response) codes have gained widespread adoption due to their ability to store information in a compact, machine-readable format. Initially developed by Denso Wave in 1994, QR codes were primarily used in the automotive industry to track parts during manufacturing. Over time, their application expanded to marketing, payments, and data sharing due to their high storage capacity and ease of scanning. A QR code typically consists of black and white modules arranged in a square grid, which can be decoded by a smartphone or QR scanner.

QR code generation involves encoding data into a format that can be translated into the grid of black-and-white modules. Various encoding schemes exist, such as numeric, alphanumeric, and binary encoding, depending on the type and amount of information. QR code generation algorithms, such as those found in libraries like **qrcodegen** (Nayuki, 2018), use error-correction techniques based on Reed-Solomon codes to ensure that the data can still be recovered even if the code is damaged. Error correction levels, which range from low to high, affect the robustness and size of the generated QR code.

In recent years, QR code generation has been improved by optimizing algorithms for faster and more efficient encoding. Research has also explored customizations, such as integrating logos or creating aesthetically pleasing QR codes, which can enhance user engagement while maintaining the code's functionality. Moreover, the integration of QR codes into modern applications, such as contactless payment systems and digital marketing, has highlighted their significance in the digital transformation era.

Overall, QR code generation is a vital technology that continues to evolve, contributing to a variety of industries by providing a quick and reliable method of linking the physical and digital worlds.

RESEARCH PLAN

This research aims to explore innovative methods for QR code generation, focusing on enhancing their functionality, security, and user experience. The primary objective is to investigate techniques for dynamic QR code creation that enable real-time data updates without altering the visual code. The study will assess different encoding algorithms to increase data storage efficiency while maintaining scan reliability. Additionally, we will evaluate security measures, such as data encryption and error correction, to prevent unauthorized access and enhance data integrity.

To understand user interaction, we will analyze the effectiveness of custom QR code designs that incorporate brand elements without compromising readability. The research will include a comparative analysis of popular QR generation libraries and tools, examining their capabilities and limitations.

The outcomes aim to provide recommendations for businesses and developers to implement advanced QR solutions that maximize data security, adaptability, and branding potential.

SL. No	Description	07/10/2024-11/10/2024	12/10/2024-16/10/2024	17/10/2024-20/10/2024	21/10/2024-29/10/2024	30/10/2024-05/11/2024	07/10/2024-10/11/2024
1.	Problem Identification						
2.	Analysis						
3.	Design						
4.	Implementation						
5.	Testing						
6.	Conclusion						

Fig. 2 Timeline chart

### Day 1: Project Initiation and planning (1 day)

- Establish the project's scope and objectives, focusing on creating an intuitive QR generation for validating the input string.
- Conduct an initial research phase to gather insights into efficient code generation and QR parsing practices.
- Identify key stakeholders and establish effective communication channels.
- Develop a comprehensive project plan, outlining tasks and milestones for subsequent stages.

### Day 2: Requirement Analysis and Design (2 days)

- Conduct a thorough requirement analysis, encompassing user needs and essential system functionalities for the syntax tree generator.
- Finalize the QR design and user interface specifications, incorporating user feedback and emphasizing usability principles.
- Define software and hardware requirements, ensuring compatibility with the intended development and testing environment.

### Day 3: Development and implementation (3 days)

- Begin coding the QR generation according to the finalized design.
- Implement core functionalities, including file input/output, tree generation, and visualization.
- Ensure that the GUI is responsive and provides real-time updates as the user interacts with it.

### Day 4: Micro controller Developing (5 days)

- Commence QR development in alignment with the finalized design and specifications.
- Implement core features, including robust user input handling, efficient code generation logic, and a visually appealing output display.
- Employ an iterative testing approach to identify and resolve potential issues promptly, ensuring the reliability and functionality of the QR.

### Day 5: Documentation, Deployment, and Feedback (1 day)

- Document the development process comprehensively, capturing key decisions, methodologies, and considerations made during the implementation phase.
- Prepare the QR generation webpage for deployment, adhering to industry best practices and standards.
- Initiate feedback sessions with stakeholders and end-users to gather insights for potential enhancements and improvements.



## METHODOLOGY


Developing a methodology for studying web generation of QR codes using C++ involves a systematic approach that encompasses theoretical understanding, practical implementation, and iterative refinement. To begin with, it's essential to establish a solid foundation in the fundamental concepts of QR code generation, including data encoding methods, error correction algorithms, and QR code structure.

1. QR Code Structure: o QR codes consist of black and white squares arranged on a grid. o They include specific patterns for alignment, version information, and error correction. o Understanding the anatomy of a QR code helps in comprehending how data is encoded and decoded.
2. Data Encoding Methods: o QR codes can encode various types of data, including numeric, alphanumeric, byte/binary, and Kanji. o Each encoding method has its own way of converting information into a format suitable for QR code structure.
3. Error Correction Algorithms: o QR codes incorporate error correction to ensure data integrity even if parts of the code are damaged.

## RESULT

### Compile the Code:

bash


 Copy code

```
g++ qr_generation.cpp -o qr_generation -lqrencode
```

**Fig 3: Compilation Of the Code**

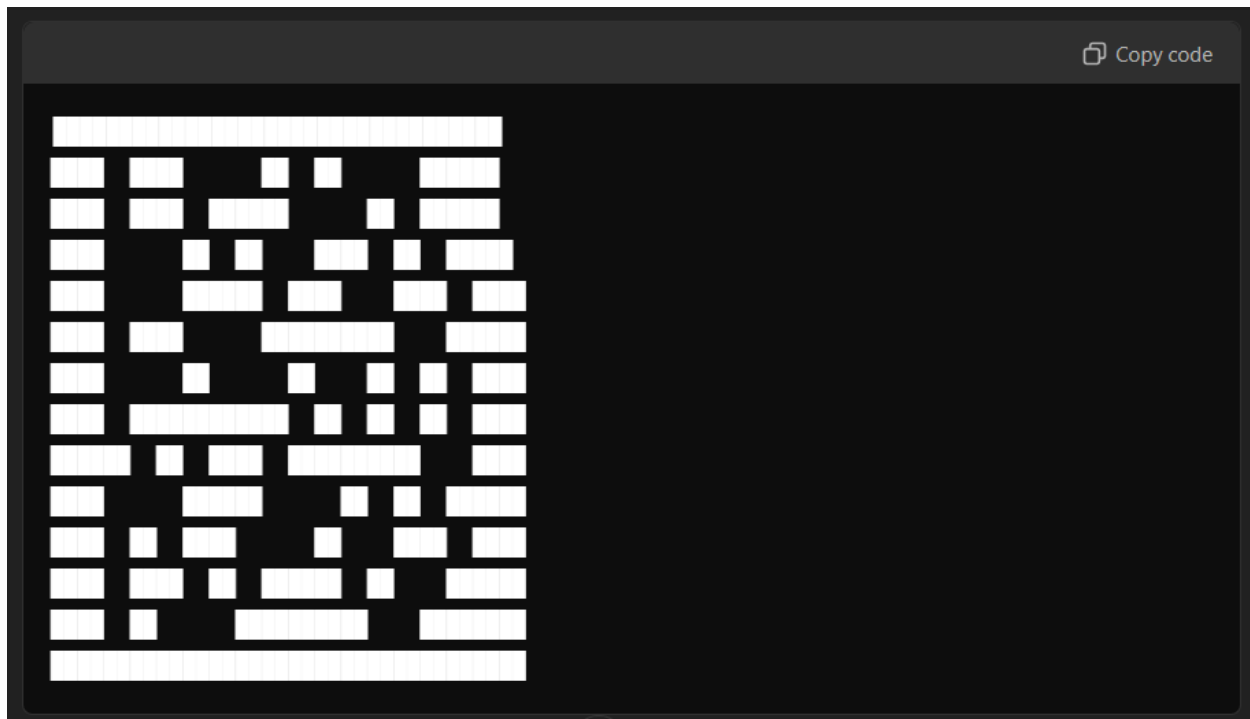
### Run the Program:

bash

 Copy code

```
./qr_generation
```

**Fig 4: Run the Program**



**Fig 5: Output of the program**

## CONCLUSION

In conclusion, studying web generation of QR codes using C++ reveals a dynamic intersection of theoretical knowledge and practical implementation essential for modern digital interactions. Mastering the foundational concepts of QR code generation, leveraging optimized algorithms, and utilizing efficient C++ libraries are fundamental steps towards achieving robust performance and reliability in web applications. By prioritizing algorithm efficiency in real time applications.

QR code generation using microcontrollers offers a practical and efficient solution for embedding QR codes into embedded systems and IoT devices. By leveraging the processing power and versatility of modern microcontrollers, developers can integrate QR code generation capabilities directly into hardware, enabling a wide range of applications, from contactless payments and product tracking to personal identification and data sharing. While microcontrollers have limited resources compared to computers or smartphones, advances in QR code algorithms and optimization techniques have made it feasible to generate QR codes with minimal computational overhead. Microcontrollers, equipped with libraries like qrcodegen or custom implementations, can handle the necessary encoding and error correction to produce functional QR codes. This capability is especially valuable in environments where compact and low-cost solutions are necessary. Overall, integrating QR code generation into microcontrollers enhances their utility and opens up new possibilities for embedded systems and real-time data communication.

## REFERENCES

- 1." Tjahyadi, Surya. "Development of QR code-based data sharing web application using system development life cycle method." *Journal of Information System and Technology (JOINT)* 2.2 (2021): 64-73.
- 2 Lee, Keun-Wang. "A study of Content Generation System using QR Code in Smart Phone Environment." *Journal of the Korea Academia-Industrial cooperation Society* 14, no. 6 (2013): 2999-3004.
- 3 Pimple, Mr. Jagdish, et al. "Qr code techniques for smart shopping: A review." *International Research Journal of Engineering and Technology (IRJET)* 5.04 (2018).
- 4 Saranya, K., R. S. Reminaa, and S. Subhitsha. "Modern applications of QR-Code for security." In *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, pp. 173-177. IEEE, 2016.
- 5 Kan, T.W., Teng, C.H. and Chen, M.Y., 2011. QR code based augmented reality applications. *Handbook of augmented reality*, pp.339-354.
- 6 Haque MS, Dybowski R. Advanced QR code-based identity card: a new era for generating student id card in developing countries. In *IEEE First International Conference on Systems Informatics, Modelling and Simulation* 2014 Apr 29 (pp. 97-103).
- 7." Venkatachalam, G., et al. "QR code generation for mall shopping guide system with security." *Asian Journal of Applied Science and Technology (AJAST)* 1.4 (2017): 37-39..
- 8 Tekawade, N., Kshirsagar, S., Sukate, S., Raut, L. and Vairagar, S., 2018, August. Social engineering solutions for the document generating generation using key-logger security mechanism and QR code. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (pp. 1-5). IEEE. 22

## SAMPLE CODE

```
#include <iostream>
#include <fstream>
#include <string>
#include "qrcodegen.hpp"

using namespace std;
using namespace qrcodegen;

// Function to save the QR code as a PPM (Portable Pixmap) image file
void saveQRCodeImage(const QRCode &qr, const string &filename) {
    const int size = qr.getSize();
    ofstream file(filename, ios::binary);

    // PPM header
    file << "P6\n" << size << " " << size << "\n255\n";

    // Write pixel data
    for (int y = 0; y < size; ++y) {
        for (int x = 0; x < size; ++x) {
            if (qr.getModule(x, y)) {
                // Black color (QR code is black on white)
                file.put(0).put(0).put(0);
            } else {
                // White color
```

```

        file.put(255).put(255).put(255);
    }
}

file.close();
cout << "QR code image saved to " << filename << endl;
}

int main() {
    string data = "https://www.example.com"; // QR code content
    bool compact = true; // Compact encoding mode (smaller QR code)
    bool verbose = false; // Display debug information

    // Generate QR code
    QrCode qr = QrCode::encodeText(data.c_str(), QrCode::Ecc::LOW);

    // Display QR code as text
    if (verbose) {
        cout << qr.toAscii() << endl;
    }

    // Save the QR code to a PPM file
    saveQRCodeImage(qr, "qrcode.ppm");

    return 0;
}

```

## OUTPUT

