FEDERAL UNIVERSITY OYE-EKITI, EKITI STATE

CSC 301 - OBJECT ORIENTED PROGRAMMING IN JAVA

DESIGN AND IMPLEMENTATION OF A QR CODE GENERATOR

Submitted to: The Department of Computer Science

By

1.	HUSSAINI-MAHA FADILAH ENEJO	CSC/2023/1089
2.	AUSTIN BLESSING OKEKENI	CSC/2023/1193`
3.	GODSON SAMUEL CHUKWUEMEKA	CSC/2023/1200
4.	OTIYOMI CALEB AKOREDE	CSC/2023/1190
5.	ABDULLAH ABDULMUIZ KOLAWOLE	CSC/2023/1209

Table of Content

INTRODUCTION

In today's digital age, Quick Response (QR) codes have become essential tools for sharing information quickly and efficiently. From marketing campaigns to contact information storage, QR codes enable seamless data transfer between physical and digital worlds. However, generating QR codes often requires users to rely on online websites or complex software, which can be inconvenient, slow, or dependent on internet connectivity.

The **QR Code Generator** was created to address this challenge by providing a standalone, user-friendly desktop application that allows users to generate, customize, and manage QR codes with ease. This project eliminates the need for external online tools and offers a fast, reliable solution for converting text, URLs, and other data into scannable QR codes.

AIM AND OBJECTIVES

Aim of the Study

The aim of this project is to develop an application that enables users to efficiently generate, customize, and manage QR codes without relying on external online tools or internet connectivity.

Objectives of the Study

i. To understand QR code technology, encoding standards, and the data capacity limits of different QR code versions.
ii. To implement features such as color customization, clipboard support, custom file saving paths, and real-time QR code preview.
iii. To test and validate the application with various data types (URLs, plain text, and long strings) to ensure reliable QR code generation.

STATEMENT OF PROBLEM

Generating QR codes has traditionally required users to rely on online web-based tools, which present several limitations and challenges. These problems include:

- **Internet Dependency**: Most QR code generators require active internet connectivity, making them inaccessible in offline environments.
- **Privacy Concerns**: Uploading sensitive data to third-party websites raises privacy and security issues.
- **Data Loss Risk**: Generated codes may not be permanently saved or may be lost if websites go down.

METHODOLOGY

This chapter presents the methodology adopted in the development of the QR Code Generator. The project follows a desktop application development approach using Java Swing for the graphical user interface and the ZXing library for QR code encoding.

Unlike projects involving Artificial Intelligence or Complex Algorithms, this application is primarily logic-driven and user input-based. However, core software development principles were applied to ensure the system is accurate, responsive, and user-friendly.

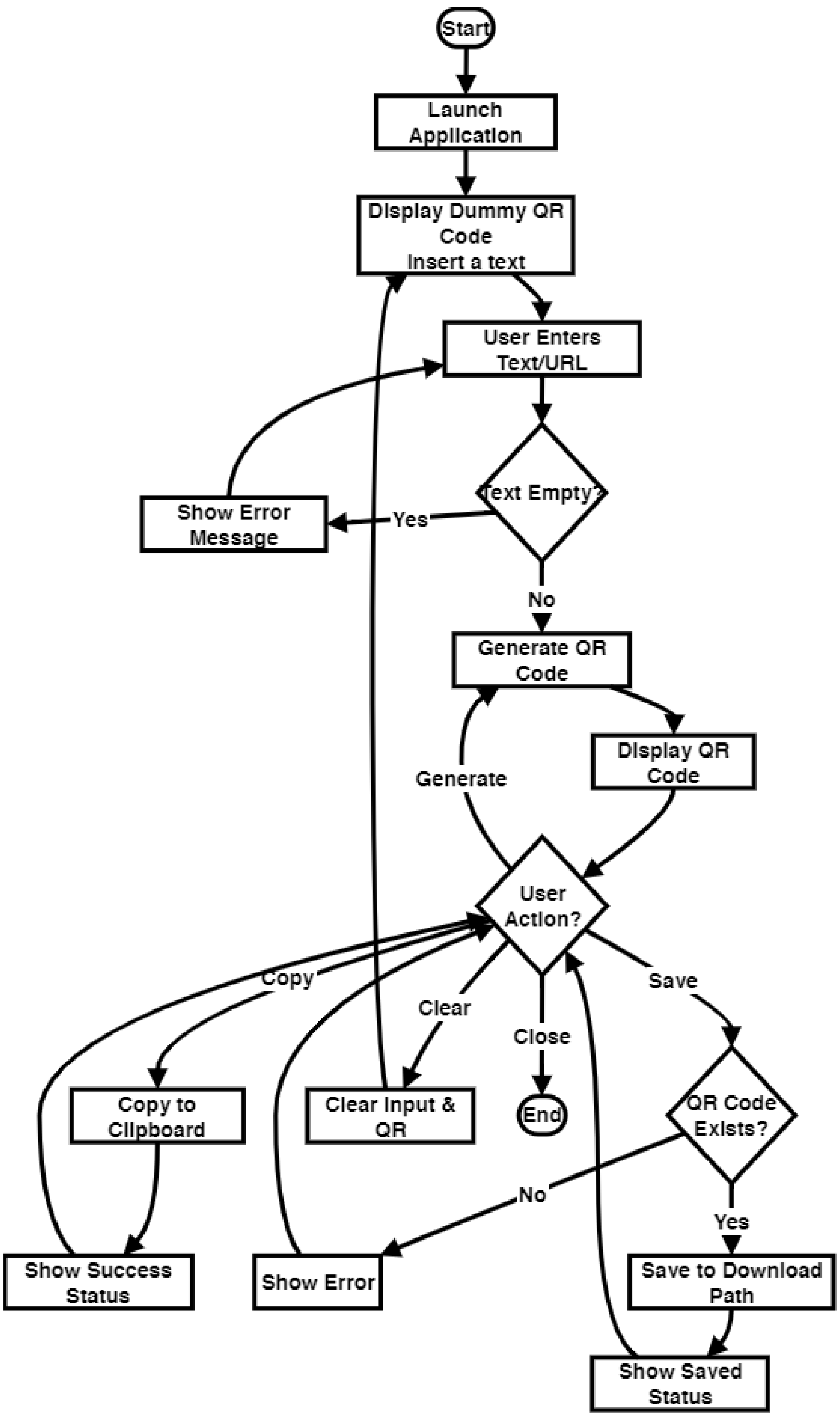
TECHNOLOGIES USED

- ➲ **Java 25**: Programming language for backend logic and frontend interface.
- ➲ **Swing Framework**: For building the graphical user interface (GUI).
- ➲ **ZXing Library**: For QR code encoding and matrix generation.
- ➲ **FlatLaf**: For modern, dark-themed UI styling.
- ➲ **Maven**: For project dependency management and build automation.

DEVELOPMENT APPROACH:

- Requirements gathering (reviewing QR code technology and user needs for offline code generation)
- Designing the input and output flow (text input, QR code display, file saving)
- Implementing the logic using Java and ZXing library.
- Testing the application with various data types (URLs, text strings, special characters)
- Packaging the application as a standalone executable JAR file for distribution.

FLOWCHART


Start
Launch Application
Display Dummy QR Code
Insert a text
User Enters Text/URL
Text Empty?
Yes
Show Error Message
No
Generate QR Code
Display QR Code
Generate
User Action?
Copy
Clear
Close
Save
Copy to Clipboard
Clear Input & QR
End
QR Code Exists?
No
Yes
Show Success Status
Show Error
Save to Download Path
Show Saved Status

System Requirements for QR Code Generator

To ensure optimal performance and compatibility with the QR Code Generator application, your system should meet the following specifications:

- **Processor:** Any modern AMD, Intel, or other compatible processor.
- **RAM:** Minimum of 512 MB installed.
- **System Type:** 64-bit operating system is recommended for optimal performance, though 32-bit systems can also run the application.
- **Operating System:** Windows 7, 8, 8.1, 10, or 11 (32-bit or 64-bit), macOS, or Linux.
- **Java Runtime:** Java 25 or higher installed.

ALGORITHM

The following are the processes involved in designing the QR Code Generator App:

Step 1: Start the application

Step 2: Application displays dummy QR code with placeholder text "Insert a text"

Step 3: User enters text or URL in the input field

Step 4: System validates input data

- If input is empty → Display error message → Return to Step 3
- If input is valid → Proceed to Step 5

Step 5: Generate QR code matrix using ZXing library

Step 6: Apply customized colors (foreground and background) to the QR code

Step 7: Display the generated QR code to the user

Step 8: User can choose to:

- Copy QR code to clipboard
- Save QR code to custom download path
- Change QR code colors and regenerate
- Clear input and start over

Step 9: If user chooses to save:

- Validate download path
- Generate timestamp-based filename
- Save QR code as PNG image
- Display success message with filename

Step 10: End or repeat from Step 3

IMPLEMENTATION

In developing the QR Code Generator Application, I started by designing and building the interface using Java Swing with Visual Studio Code as my code editor. Before starting the design, I installed Java 25 and configured Maven for dependency management. I then used the ZXing library to enable the core functionalities including QR code encoding algorithms and color customization features.

Application Interface Sections

Title Section: At the top of the application, I placed a bold, centered title "QR Code Generator" using a 26pt font to clearly identify the purpose of the application.

Input Panel: Below the title, I designed a rounded input panel where users can enter their text or URL. This section includes a labeled text field with a green accent border, allowing real-time QR code generation as users type. The input field has a dark background with white text for better readability.

Color Customization Section: I implemented two color picker buttons labeled "QR Color" and "Background" that allow users to customize the foreground and background colors of their generated QR codes. This feature provides flexibility for users who want personalized QR codes matching their branding.

Action Buttons: I created five main action buttons:

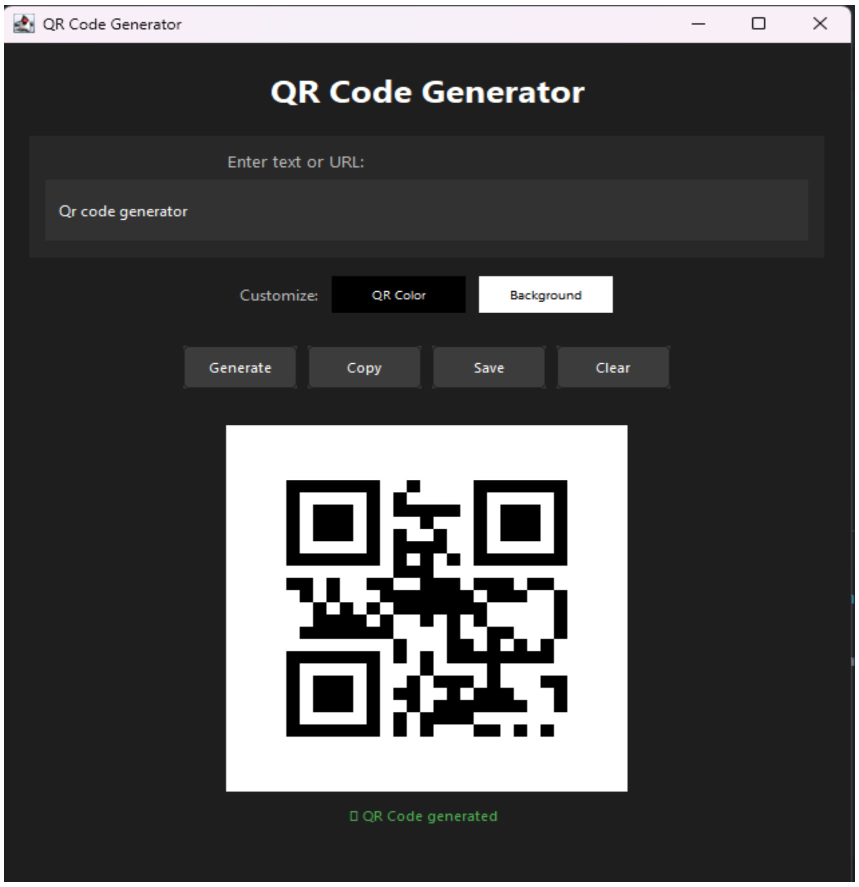
- **Generate**: Manually triggers QR code generation from the input text
- **Copy**: Copies the generated QR code image directly to the system clipboard
- **Save**: Saves the QR code as a PNG file to a selected directory
- **Clear**: Resets all inputs and returns to the dummy QR code display

QR Code Display: I positioned a 300x300 pixel label in the center of the application with a rounded green border to display the generated QR code. On startup, this section shows a dummy QR code with the text "Insert a text" to guide users.

Status Panel: At the bottom, I added a status display that provides real-time feedback to users. It shows operation results (generation success, copy confirmation, save confirmation, etc.) and displays the current download path. Status messages use color coding: green for success, red for errors, and orange for warnings.

Responsive Design: I implemented a scrollable interface that adapts to different window sizes (minimum 550x750 pixels), ensuring the application remains functional across various screen resolutions. I used FlatLaf for modern dark theme styling and added rounded borders throughout the interface for a contemporary look.

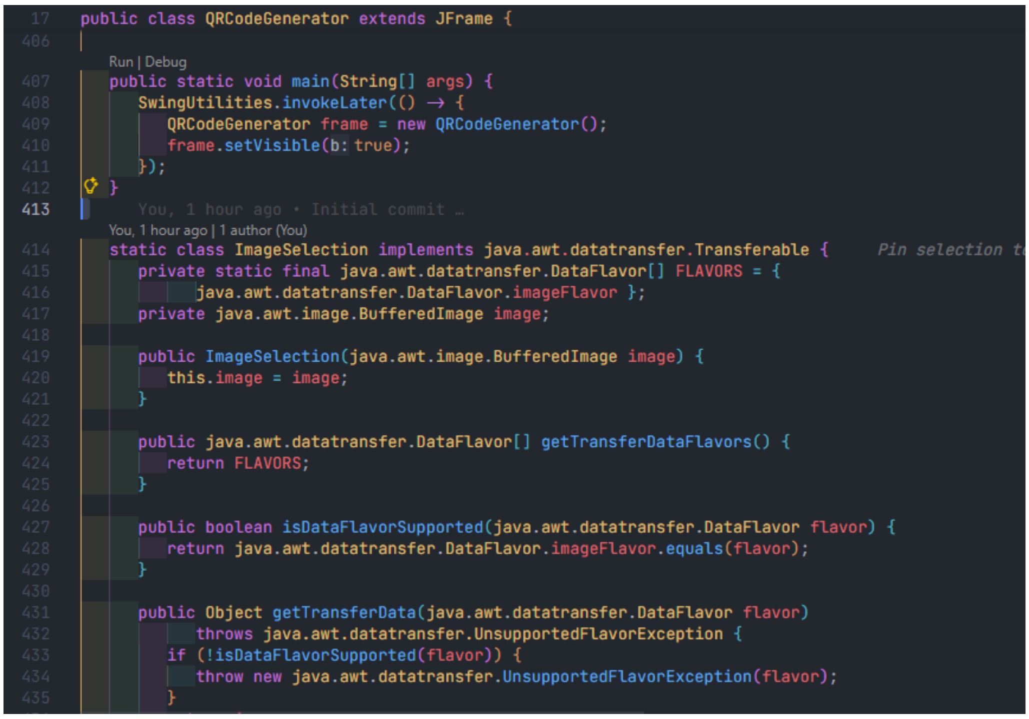
PREVIEW



Qr code generator

□ QR Code generated

CODE SNIPPET



```
public class QRCodeGenerator extends JFrame {

    Run | Debug
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            QRCodeGenerator frame = new QRCodeGenerator();
            frame.setVisible(b: true);
        });
    }
        You, 1 hour ago • Initial commit …
    You, 1 hour ago | 1 author (You)
    static class ImageSelection implements java.awt.datatransfer.Transferable {      Pin selection t
        private static final java.awt.datatransfer.DataFlavor[] FLAVORS = {
                java.awt.datatransfer.DataFlavor.imageFlavor };
        private java.awt.image.BufferedImage image;

        public ImageSelection(java.awt.image.BufferedImage image) {
            this.image = image;
        }

        public java.awt.datatransfer.DataFlavor[] getTransferDataFlavors() {
            return FLAVORS;
        }

        public boolean isDataFlavorSupported(java.awt.datatransfer.DataFlavor flavor) {
            return java.awt.datatransfer.DataFlavor.imageFlavor.equals(flavor);
        }

        public Object getTransferData(java.awt.datatransfer.DataFlavor flavor)
                throws java.awt.datatransfer.UnsupportedFlavorException {
            if (!isDataFlavorSupported(flavor)) {
                throw new java.awt.datatransfer.UnsupportedFlavorException(flavor);
            }
```

USER GUIDE

How to use the QR Code Generator Application

To run this project, you must have Java 25 or higher installed on your PC with an operating system that supports Java (Windows, macOS, or Linux).

After launching the QR Code Generator application, you must follow these steps:

Step 1: Launch the QR Code Generator application by double-clicking the executable JAR file or running it from the command line

Step 2: The application will display a window with a dummy QR code showing the placeholder text "Insert a text"

Step 3: Locate the text input field at the top of the application and enter your desired text, URL, or data

Step 4: The QR code will automatically generate and display in real-time as you type

Step 5: (Optional) To customize the QR code colors:

- Click the "QR Color" button to change the QR code pattern color
- Click the "Background" button to change the background color
- The QR code will automatically update with your selected colors

Step 6: After generating your QR code, you can perform the following actions:

- Click "Copy" to copy the QR code image to your clipboard for pasting into other applications
- Click "Save" to save the QR code as a PNG image file
- Click "Clear" to reset all inputs and start over

Step 7: If you choose to save the QR code:

- Select your desired download folder using the file chooser dialog
- The application will automatically generate a timestamped filename
- Your QR code will be saved successfully with a confirmation message

Step 8: Review the status messages at the bottom of the window for feedback on your actions (generation, copying, saving, etc.)

Step 9: You can generate multiple QR codes in a single session by repeating steps 3-7

Step 10: Close the application when finished

Note: The application works completely offline. Ensure your input text is not empty before generating a QR code for accurate results.

CONCLUSION

The QR Code Generator application provides a simple and efficient solution for users to generate customized QR codes without relying on external online tools. By automating the conversion of text and URLs into scannable QR codes, the application eliminates the need for internet connectivity and third-party services.

Its user-friendly interface, customizable colors, clipboard support, and flexible file management ensure a seamless user experience. The real-time QR code generation provides instant visual feedback, while the responsive design adapts to different window sizes for accessibility across various systems.

This tool serves as a valuable resource for businesses, educators, event organizers, and individuals who need to generate QR codes quickly and reliably. The application demonstrates practical desktop application development using modern Java technologies and represents the shift toward standalone, offline-capable digital solutions that give users full control over their data and workflow.

REFERENCE

- ZXing (Zebra Crossing) Library - https://github.com/zxing/zxing
- FlatLaf - Modern Java Swing Look and Feel - https://www.formdev.com/flatlaf/
- Oracle Java Documentation - https://docs.oracle.com/javase/
- QR Code Standards - ISO/IEC 18004:2015 - https://www.iso.org/standard/62021.html
- Apache Maven - https://maven.apache.org/
- Wikipedia - QR Code - https://en.wikipedia.org/wiki/QR_code