# **FlashQR: Instant Code at Your Fingertips**

In today's digital world, seamless information sharing is essential. Our project, **"Qreate: The Smart QR Generator,"** is a web-based application that dynamically generates QR codes from user-provided text. This tool simplifies data encoding, enabling users to convert plain text, URLs, or any other information into a scannable QR code within seconds.

Our system is built using **MongoDB, Server.js, HTML, CSS, and JavaScript**, ensuring a robust, scalable, and user-friendly experience. The back-end, powered by **Node.js with Express.js**, efficiently processes user inputs, stores QR code generation history in **MongoDB**, and delivers real-time responses. The front-end, crafted with **HTML, CSS, and JavaScript**, offers an interactive and intuitive interface for users to generate, preview, and download their QR codes effortlessly.This project finds applications in various domains, including **business promotions, contact-less payments, event ticketing, authentication systems, and personal digital branding**. By integrating advanced web technologies, **Qreate** ensures fast, reliable, and secure QR code generation, making information sharing more efficient than ever.

### ****Project Structure: FlashQR – Instant Code at Your Fingertips****

* **FlashQR: Instant Code at Your Fingertips/** – A streamlined solution for generating and managing QR codes with ease!
* **public/** – The front-end powerhouse!
* **index.html** – The interactive interface for QR code generation.
* **style.css** – The aesthetic backbone, ensuring a sleek and user-friendly design.
* **app.js** – The dynamic script handling user input, form validation, and API interactions.
* **server.js** – The command center! This Node.js-powered back-end efficiently processes requests, generates QR codes, and stores data securely in MongoDB.

With **FlashQR**, QR code creation is just a scan away!….

**Architecture Overview**

**+-------------+ +--------------+ +-----------------+**

**| User | ----> | Frontend | ----> | Backend (API)**

**+-------------+ +--------------+ +-----------------+**

**| |**

**V V**

**+------------------+ +-----------------+**

**|QR Code Gen| | Database |**

**| (JavaScript) | | (MongoDB) |**

**+-------------------+ +------------------+**

**Explanation of the Architecture:**

The **FlashQR: Instant Code at Your Fingertips** project follows a **3-Tier Architecture**, separating the front-end, back-end, and database to ensure scalability, maintainability, and a smooth user experience.

### ****1. Frontend (Client-Side)****

**Technology:** HTML, CSS, JavaScript

* **User Interaction:** The user interacts with the **UI** (built using HTML, styled with CSS). They enter text into a form, which is used to generate a QR code.
* **Form Validation & API Calls:** **JavaScript (app.js)** validates the form and sends the data (QR code text) to the back-end via a **POST request**.
* **QR Display:** Once the QR code is generated, it is displayed on the page for the user to view or save.

### ****Backend (Server-Side)****

* **Technology:** Node.js, Express.js
* **API Handling:** The back-end server, built with **Express.js**, listens for API requests from the front-end. When the front-end sends data (QR code text), the back-end processes it.
* **QR Code Generation:** The back-end handles the logic for generating the QR code (using libraries like qr-code or similar) and sends the generated code back to the front-end.
* **Data Storage:** The server also stores the generated QR codes in **MongoDB** for future retrieval and reference.

### ****Database (MongoDB)****

* **Technology:** MongoDB (Cloud - MongoDB Atlas)
* **Data Storage:** The back-end saves the **QR code data** (text and image) in the MongoDB database. This allows users to store QR codes they generate and retrieve them later if needed.
* **Efficient Querying:** MongoDB allows fast storage and retrieval of QR codes, ensuring a smooth experience for users accessing saved codes.

### ****Data Flow in the System****

1. **User submits text** for QR code generation in the front-end.
2. The front-end **sends the data** to the back-end API to process the QR code generation.
3. **Back-end generates the QR code** and sends it back to the front-end.
4. **User views the generated QR** and can choose to save it, triggering a request to store it in **MongoDB.**
5. **Saved QR codes** can be retrieved and displayed as needed.

## **Frontend – HTML & CSS**

## **public/index.html**

<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="UTF-8">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>QR code generator</title>

        <link rel="stylesheet" href="styles.css">

        <script src="https://cdnjs.cloudflare.com/ajax/libs/qrious/4.0.2/qrious.min.js"></script>

    </head>

    <body>

        <div class="container">

            <h2>QR code genarator</h2>

            <form id="qrForm">

                <input type="text" id="qrText" place holder="enter text for QR code" required>

                <button type="submit">Generate QR code</button>

            </form>

            <div id="qrCodeContainer">

                <canvas id="qrCanvas"></canvas>

            </div>

            <button id="saveQR" style="display: none;">Save QR code</button>

        </div>

        <script src="script.js"></script>

    </body>

</html>

### ****Explanation of the Code****

This is the **HTML structure** for a simple **QR code generator**:

### ****Head Section :****

* <meta charset="UTF-8">: Ensures proper encoding of the document.
* <meta name="viewport" content="width=device-width, initial-scale=1.0">: Makes the page mobile responsive.
* <title>QR code generator</title>: The title of the web-page.
* <link rel="stylesheet" href="styles.css">: Links to an external **CSS** file for styling.
* <script src="https://cdnjs.cloudflare.com/ajax/libs/qrious/4.0.2/qrious.min.js"></script>: Includes the **QRious** library (used for QR code generation).

### ****Body Section:****

* <div class="container">: A wrapper for the content to center it and apply styles.
* <h2>: Title for the QR code generator.
* <form id="qrForm">: Form that accepts input from the user to generate a QR code.
* <input type="text" id="qrText" placeholder="enter text for QR code" required>: Text input for the user to enter data to encode into the QR code.
* <button type="submit">Generate QR code</button>: Button to trigger the QR code generation.
* <div id="qrCodeContainer">: A container for displaying the generated QR code.
* <canvas id="qrCanvas"></canvas>: A canvas element where the QR code will be drawn
* <button id="saveQR" style="display: none;">Save QR code</button>: A button that will appear after the QR code is generated, allowing the user to save it.

### ****Script Section :****

* <script src="script.js"></script>: Links to an external **JavaScript** file (**script.js**) that will handle form submission, QR code generation, and saving the QR code.

Overall Functionality :

1. **User Input**: The user enters text into the input field.
2. **Generate QR Code**: On clicking the "Generate QR code" button, the QR code will be generated using the **QRious** library and displayed in the <canvas> element.
3. **Save Button**: After generating the QR code, a "Save QR code" button will appear, which will allow the user to save the generated QR code.

## **public/style.css**

**body {**

**font-family: Arial, sans-serif;**

**background-color: #f4f4f4;**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**height: 100vh;**

**margin: 0;**

**flex-direction: column;**

**}**

**/\* Container \*/**

**.container {**

**background: white;**

**padding: 20px;**

**border-radius: 10px;**

**box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);**

**text-align: center;**

**width: 450px;**

**}**

**/\* Form Inputs \*/**

**input {**

**width: 100%;**

**padding: 10px;**

**margin: 10px 0;**

**border: 1px solid #ddd;**

**border-radius: 5px;**

**font-size: 16px;**

**}**

**/\* Button \*/**

**button {**

**background: #007bff;**

**color: white;**

**border: none;**

**padding: 10px;**

**font-size: 16px;**

**border-radius: 5px;**

**cursor: pointer;**

**width: 100%;**

**margin-top: 10px;**

**}**

**button:hover {**

**background: #0056b3;**

**}**

**/\* QR Code Display \*/**

**#qrCodeContainer {**

**margin-top: 20px;**

**}**

### ****Explanation of the Code****

1. **Body** :

* Uses **Arial** font.
* Centers content both vertically and horizontally with **flexbox**.
* Sets a light **background color** and ensures the page takes up the full height of the viewport

1. **Container :**

* White background with **padding** and **rounded corners**.
* Adds a subtle **box-shadow** for depth.
* Centers text and limits the width to **450px**.

1. **Form Inputs :**

* Input fields stretch to **100% width** and have padding, rounded borders, and a slightly larger font size.
* Margins between the input fields for spacing.

1. **Button :**

* Blue background with white text, **rounded corners**, and **full width**.
* **Hover effect**: Changes the background color to a darker blue when hovered.

1. **QR Code Display** :

* Adds **margin-top** to the container holding the QR code.