**“QR CODE SCANNER & GENERATOR”**

Real Time Research Project

Bachelor of Technology (B.Tech – II Year / II Semester)

in

Computer Science and Engineering

By

**R. Satya Hemesh - 23AG1A05B2**

Under the Esteemed Guidance of

**Dr. M. V. Vijaya Saradhi**

**Professor & Dean of CSE Department.**



Department of Computer Science and Engineering

**ACE Engineering College**

An AUTONOMOUS Institution

NBA Accredited B. Tech Courses, Accorded NAAC 'A' Grade

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, Telangana)

Ankushapur(V), Ghatkesar(M), Medchal – Malkajgiri Dist. - 501 301.

June 2025

 ACE Engineering College

An AUTONOMOUS Institution

Website: www.aceec.ac.in E-mail: info@aceec.ac.in

Department of Computer Science and Engineering

Certificate

This is to certify that the Real Time project work entitled **"QR Code Generator & Scanner"** submitted by **R. Satya Hemesh (23AG1A05B2),** in partial fulfillment of Real Time Project work during the academic year 2024-25 is a record of Bonafide work carried out by them under our guidance and supervision.

Dr. M. V. Vijaya Saradhi Mr. B. R. Srinivasa Rao

Project Guide Assistant Professor &

Department of CSE Project Coordinator

Dr. V. Ravi Kumar Dr. M. V. Vijaya Saradhi

Head of Section - B Professor and Dean

Assistant Professor Department of CSE

**Acknowledgement**

We would like to express our gratitude to all the people behind the screen who have helped us transform an idea into a real time application.

We would like to express our heart-felt gratitude to our parents without whom we would not have been privileged to achieve and fulfill our dreams.

A special thanks to our General Secretary, **Prof. Y. V. Gopala Krishna Murthy**, for having founded such an esteemed institution. Sincere thanks to our Joint Secretary

Our Sincere thanks to our COO **Mr. Y.V.Raghu Vamshi**, for his support in doing project work.

We are Deeply grateful to our beloved Principal, **Dr. K.S. Rao** for permitting us to carry out this project.

We are very much thankful to our beloved Vice Principal & Dean-Skill Development **Dr. M. Murali** for his continuous encouragement to fulfill our project.

We are profoundly grateful to **Dr. M. V. Vijaya Saradhi**, Professor and Dean, Department of Computer Science and Engineering, for being an inspirational Leader, Excellent Guide and also for His continuous support throughout our Project Work.

We are very grateful to our **Dr. V. Ravi Kumar – Assistant Professor – Head of Section - B** for His continuous support towards completion of our Project Work.

We are extremely thankful to **Mr. B R Srinivasa Rao**, Assistant Professor and Project Coordinator, who helped us throughout our Real-Time / Field Based Research Project.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, We would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased our task.

R. Satya Hemesh (23AG1A05B2)

P. Sri Lakshmi (23AG1A05A9)

Mohammed Nowfa (23AG1A0599)

**ACE Engineering College  
An Autonomous Institution**

Ghatkesar, Medchal (Dist.), Hyderabad, Telangana State – 501301(NBA Accredited B.Tech Courses, Accorded NAAC 'A' Grade) **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
REAL TIME RESEARCH PROJECT  
INDEX**

**CONTENTS PAGE NO.**

**Certificate 2**

**Acknowledgement 3**

**Abstract 6**

1. **Introduction 7-15**
   1. Purpose
   2. Scope
   3. Real-Time Usage & Applications
   4. Target Audience

1.5 Overall Description

1.6 User Interface

1.7 System Interface

1.8 API / Library Used

1. **System Analysis 16-18**

2.1 Existing System Drawbacks

2.2 Proposed System

2.3 Team Size

1. **Requirements 19**

3.1 Software Requirements

3.2 Hardware Requirements

1. **Architecture Diagrams 20-23**

4.1 Use Case Diagram

4.2 Class Diagram

4.3 Sequence Diagram

4.4 Collaboration Diagram

4.5 Activity Diagram

4.6 Component Diagram

4.7 Deployment Diagram

4.8 Entity – Relationship Diagram

1. **Software Design 24-25**

5.1 Architectural Design

5.2 Design Modules

5.3 Data Flow Summary

5.4 Layered Design View

5.5 Design Features

5.6 Future Enhancements & Design Road-Map

1. **Module Description 26-27**

6.1 QR Code Generator Module

6.2 QR Code Scanner Module

6.3 User Interface ( UI ) Module

1. **Implementation 28-34**

7.1 Source Code: Integrated HTML Page

1. **Output Screens 35**
2. **Testing 36-38**

9.1 Testing Objectives

9.2 Test Cases

9.3 Compatibility Testing

9.4 Responsive UI Testing

9.5 Error Handling

9.6 Performance Testing

9.7 Usability & Accessibility

1. **Conclusion 39**
2. **Future Enhancements 40-41**

11.1 Real-Time Camera Based Testing

11.2 Custom QR Code Styling

11.3 Scan History

11.4 QR Code Support Format

11.5 Accessibility Features

11.6 Multi-Lingual Support

11.7 Progressive Web App [ PWA ]

11.8 Backend Integration

11.9 Error Correction Customization

11.10 Security Enhancements

1. **References 42**

**ABSTRACT**

The QR Code Generator and Scanner is a web-based tool designed to create and decode QR codes efficiently. QR codes are widely used for storing and sharing information such as URLs, contact details, and text in a scannable format. This project provides a fast, user-friendly, and responsive solution for generating and scanning QR codes using HTML, CSS, and JavaScript without external dependencies. The QR Generator allows users to input text or links and instantly convert them into QR codes, which can be downloaded and shared. The QR Scanner enables users to scan QR codes using a device camera or by uploading an image to extract the embedded data. The application is lightweight and works seamlessly across devices, making it accessible and convenient. With applications in business, marketing, education, and personal use, this project enhances data exchange and accessibility.

Future improvements may include custom QR code designs, support for various data formats, and offline functionality. This project showcases the practicality and efficiency of QR technology, providing a simple yet powerful tool for modern digital communication.

**1. INTRODUCTION**

The main purpose of this project is to generate and to scan the QR codes for storing and retrieving the information efficiently. The QR Code Generator and Scanner is a web-based tool that allows users to create and scan QR codes efficiently. It helps store and share information such as URLs, the contact details, and text in a scannable format. This project aims to provide a user-friendly, fast, and responsive solution using HTML, CSS, and JavaScript. This eliminates the need for third-party applications and ensures a lightweight, efficient, and privacy-focused solution. Current online QR code generators allow users to create QR codes but come with several limitations such as limited customization, paid features, and dependency on an internet connection. Mobile-based QR code scanners, like Google Lens, allow users to scan QR codes instantly but rely on third-party applications and may include ads or require special permissions.

1.1 PURPOSE

This project aims to develop a lightweight, dependency-free web application for QR code generation and scanning. Unlike existing tools that rely on external software, internet connectivity, or third-party services, our solution is built entirely with HTML, CSS, and JavaScript, ensuring full offline functionality, enhanced performance, and cross-device compatibility. The tool features a clean, user-friendly interface and processes all data client-side, prioritizing user privacy and security. By eliminating ads, installations, and hidden costs, this application offers a seamless, accessible experience for diverse users across domains like education, healthcare, and logistics—demonstrating the power of modern web technologies in secure and efficient data sharing.

1.2 SCOPE

This project focuses on the complete development of a browser-based QR code generator and scanner that is fully client-side and requires no installation. Designed for secure and fast data sharing, the tool allows users to generate QR codes from text or URLs, download them as images, and scan existing QR codes using a device camera or image upload. All processing is done locally, ensuring user privacy and eliminating reliance on external servers. The application works offline after the initial load and is optimized for responsiveness across all devices, including smartphones and desktops. Its clean interface and compliance with accessibility standards make it easy to use for all user groups. Future enhancements may include customizable QR styles (colours, frames, logos), support for various data formats like vCard’s and Wi-Fi credentials, scan history stored locally, and optional session management without cloud integration.

The scope ensures a simple, secure, and efficient tool suitable for both personal and institutional use.

1.3 REAL TIME USAGE & APPLICATIONS

QR codes have become a ubiquitous part of modern digital communication, facilitating fast, contactless, and reliable data sharing. The web-based QR Code Generator and Scanner developed in this project can be effectively utilized in various real-time applications across multiple sectors. Here are some significant real-time usage scenarios and applications tailored to the functionality and objectives of your project:

**1. Contactless Business Cards**

Professionals can generate QR codes that encode their contact information (name, email, phone number, LinkedIn profiles, etc.), allowing others to scan and save the details directly to their devices—minimizing the need for physical cards and enabling efficient digital networking.

**2. Event Check-ins and Registrations**

Organizers of events (such as seminars, workshops, or tech fests) can create QR codes linked to registration pages or participant details. Attendees can simply scan the code at the entry point for a quick, contactless check-in experience.

**3. Wi-Fi Credential Sharing**

Users can encode Wi-Fi credentials (SSID and password) into a QR code. Scanning the code allows guests to connect to the network without needing to type long passwords—enhancing both security and convenience.

**4. Educational Resources**

Teachers and institutions can use QR codes to share assignment links, video lectures, PDFs, or websites with students. Students can scan the QR code displayed in a classroom or presentation to access learning material instantly.

**5. Marketing and Advertising**

Businesses can print QR codes on posters, flyers, and packaging that link to promotional websites, discount coupons, or product videos—providing a bridge between offline media and digital content.

**6. Healthcare Information Sharing**

Hospitals and clinics can use QR codes to store patient IDs, test results, or digital prescriptions for secure and quick access by authorized personnel using mobile devices.

**7. Personal Use**

Individuals can generate QR codes to share social media profiles, contact lists, or even text notes. The scanner functionality also allows users to extract data from QR codes encountered in day-to-day life.

**8. Inventory and Logistics Management**

Though lightweight, the tool can assist in encoding product IDs or shipment data for scanning during stock handling, offering a simple solution for small-scale logistics operations.

**9. Authentication & Secure Access**

With slight modifications, the QR scanner can be adapted for use in two-factor authentication (2FA) workflows or secure building access through QR badges.

1.4 TARGET AUDIENCE

**1. General Public / Individual Users**

* Users who want to generate QR codes to share their Wi-Fi passwords, personal contact details, or links to their social media profiles.
* Anyone looking to scan QR codes found in public places such as on packaging, posters, or product labels without downloading additional apps.

**2. Small Business Owners & Entrepreneurs**

* Shopkeepers, freelancers, or small-scale service providers who need QR codes for payment links, service menus, or product catalogues.
* Businesses looking for a free, privacy-focused alternative to third-party commercial QR tools.

**3. Educational Institutions**

* Teachers and students who want to quickly share assignments, learning materials, video links, or online tests via QR codes.
* Schools and colleges aiming to use QR codes for event management, attendance tracking, or feedback collection.

**4. Event Organizers & Administrators**

* Professionals handling conferences, workshops, or community events who use QR codes for participant registration, agenda sharing, or entry validation.
* Institutions that require scalable, touch-free check-in systems.

**5. Developers & Tech Enthusiasts**

* Developers looking for open-source or customizable tools that can be integrated into other systems (e.g., websites, kiosks).
* Students and hobbyists learning web development or data security concepts through QR technology.

**6. Corporate & Office Environments**

* Employees generating QR codes for quick access to shared documents, calendars, and internal tools.
* Organizations using QR codes for internal workflows, contactless ID systems, or Wi-Fi onboarding.

**7. Marketing & Advertising Professionals**

* Agencies using QR codes for campaigns, feedback forms, discount coupons, and product landing pages in both digital and print media.
* Creative professionals embedding QR codes into posters, banners, or presentations.

**8. Healthcare Providers**

* Clinics and pharmacies who wish to provide patients with scannable prescriptions, health record access, or appointment scheduling links.

1.5 OVERALL DESCRIPTION

The QR Code Generator and Scanner is a browser-based web application designed to provide users with a seamless, lightweight solution for creating and decoding QR codes without requiring installations or external server support. This application operates entirely on the client-side, using standard web technologies like HTML, CSS, and JavaScript, ensuring maximum compatibility and user privacy.

**The main functionalities of the system include:**

* QR Code Generation: Converts user-entered data such as text or URLs into downloadable QR code images.
* QR Code Scanning: Scans QR codes using the device's camera or uploaded image files and displays the decoded result instantly.

The application supports real-time usage in environments such as business transactions, event check-ins, educational material distribution, and personal data sharing. Designed to work across all modern web browsers and devices, it emphasizes accessibility, privacy, and ease of use. The tool can also function offline after the initial load, which enhances its utility in low-connectivity areas.

1.6 USER INTERFACE

The User Interface (UI) of the application is developed to be clean, responsive, and user-friendly. It is designed for users with minimal technical expertise while offering efficiency and clarity for experienced users as well.

**Key UI Components:**

1. **Home Screen**  
   A central screen with clear buttons to choose between generating or scanning a QR code.
2. **QR Generator Section**
   * Input text area for data to be encoded.
   * "Generate" button to create the QR.
   * Display panel for the generated QR code.
   * "Download QR" button for saving the image locally.
3. **QR Scanner Section**
   * Camera preview using device camera.
   * "Start Scanning" and "Stop" buttons.
   * Option to upload a QR image for scanning.
   * Display area for decoded data.
4. **Responsive Design**
   * Adapts layout for both mobile and desktop screens.
   * Icons and buttons optimized for touch and mouse interaction.
5. **Error and Status Messages**

Visual feedback on success, failure, or invalid inputs.

1.7 SYSTEM INTERFACE

The system is designed to run completely on the frontend, eliminating the need for backend APIs or server-based processing. It interacts directly with:

* **Browser Rendering Engine** for displaying the UI
* **Camera Access API** (WebRTC/MediaDevices API) for real-time scanning
* **DOM Events and Handlers** for input and user interactions
* **File Reader API** for processing uploaded QR images

The application architecture includes:

* **Input Layer**: Handles text input and camera/image selection
* **Processing Layer**: Generates or decodes QR code using JavaScript
* **Output Layer**: Displays QR images and scanned content

The structure enables low latency operations and minimal consumption, making it efficient on both high-end & entry-level devices

1.8 API / LIBRARY USED

To implement core functionalities of QR code generation and scanning, the following JavaScript libraries and browser APIs are used:

* **qrcode.js**
* **Purpose**: To generate QR code images from user input
* **Usage**: Converts text into a QR image displayed on the page
* **Advantages**: Lightweight, fast, works offline
* **html5-qrcode or js-QR**
* **Purpose**: To decode QR codes using the device camera or uploaded image
* **Usage**: Accesses camera feed, detects QR codes in real time, and extracts the encoded content
* **Advantages**: Pure JavaScript, no server-side decoding, good accuracy
* **Web APIs Used**
* **Media Devices API (WebRTC)** – For accessing the camera
* **Canvas API** – For rendering and analysing QR images
* **File Reader API** – For scanning QR codes from uploaded images
* **Blob & URL APIs** – For downloading QR code images

These libraries and APIs are chosen specifically for their compatibility, simplicity, and ability to operate fully in-browser with no dependency on external services or databases.

**2. SYSTEM ANALYSIS**

2.1 EXISTING + SYSTEM & DRAWBACKS

In the current technological landscape, QR codes are widely used in various domains such as marketing, payments, authentication, and education. However, most of the existing QR code generator and scanner applications come with significant limitations, especially in terms of usability, accessibility, privacy, and platform dependency. Below are some common drawbacks of existing systems:

**1. Dependency on External Applications**

Most QR code tools are available as downloadable mobile or desktop applications. This requires users to install additional software, which can be inconvenient or impossible in restricted environments (e.g., schools, workplaces, public systems).

**2. Internet Dependency**

Many QR scanners or generators require continuous internet connectivity because they rely on server-side processing or cloud-based services. This makes them unusable in low-connectivity or offline scenarios.

**3. Privacy and Data Security Risks**

Several existing tools transmit the data to remote servers for processing, which poses serious privacy concerns—especially when QR codes are used for sensitive information like payments, authentication, or healthcare data.

**4. Heavy and Resource-Intensive**

Apps and web services with advanced features often use large frameworks or have unnecessary dependencies, making them slow to load and heavy on system resources, especially on low-end or mobile devices.

**5. Limited Platform Support**

Some QR solutions are platform-specific (Android-only or iOS-only), limiting accessibility for users on different operating systems. Desktop users are often excluded or provided with less optimized experiences.

**6. Advertisements and Hidden Costs**

Many free QR code apps inject ads, limit features in the free version, or watermark the output. Users must upgrade to paid versions for full access, making them unsuitable for educational or public-use cases.

**7. Poor User Experience**

Cluttered UI, non-intuitive controls, or a lack of clear feedback are common issues in existing applications, especially in scanner tools. This affects accessibility for non-technical or elderly users.

**8. No Offline Support**

Most tools fail to function without internet access. This is problematic in fieldwork, remote locations, or environments where offline tools are essential for productivity.

**9. Limited Customization and Control**

Users are rarely allowed to customize QR codes or fine-tune scanning options. Some apps force template use or restrict input types (e.g., text only, no file links or vCards).

2.2 PROPOSED SYSTEM

The proposed **QR Code Generator and Scanner** addresses all these issues by providing a:

* **Browser-Based Solution**  
  Runs directly in the browser using HTML, CSS, and JavaScript—no installation required.
* **Fully Client-Side Functionality**  
  All QR generation and scanning are done within the browser. No data is sent to external servers, preserving user privacy.
* **Offline Operation Support**  
  Once loaded, the app can generate and scan QR codes even without an internet connection.
* **Responsive and Clean UI**  
  Optimized for both mobile and desktop devices using CSS responsiveness and clear interface structure.
* **Open-Source Libraries**  
  Utilizes lightweight, open-source JavaScript libraries (qrcode.js, html5-qrcode) to perform core tasks efficiently.

2.3 TEAM SIZE

The project was undertaken by a team of three undergraduate students from the Department of Computer Science and Engineering at ACE Engineering College.

**Team Members:**

* **R. Satya Hemesh (23AG1A05B2)**
* **Mohammed Nowfa (23AG1A0599)**
* **P. Sri Lakshmi (23AG1A05A9)**

Each team member contributed actively to the successful completion of the project. The responsibilities were collaboratively shared across the following areas:

* **Frontend Design and UI Development:**  
  Designing a clean and responsive interface using HTML and CSS.
* **QR Code Generation and Scanning Logic:**  
  Implementing JavaScript-based libraries for real-time QR code generation and decoding.
* **Documentation and Presentation:**  
  Preparing the formal documentation, project report, and visual presentation content.

The team worked collectively under the **Guidance of Dr. M.V. Vijaya Saradhi**, who provided continuous support and valuable feedback throughout the project lifecycle.

**3. REQUIREMNTS**

3.1 SOFTWARE REQUIREMENTS

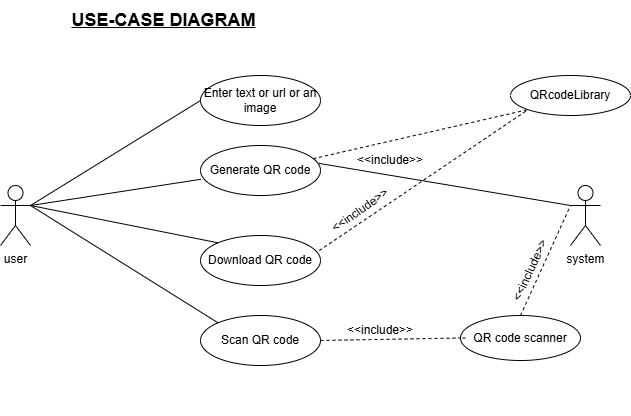
|  |  |
| --- | --- |
| **Component** | **Details** |
| **Operating System** | Windows 10 or higher / Linux / macOS / Android (browser-based) |
| **Browser** | Google Chrome, Mozilla Firefox, Microsoft Edge, Safari |
| **Languages Used** | HTML, CSS, JavaScript |
| **Editor** | Visual Studio Code, Sublime Text, or any text editor |
| **Libraries/Frameworks** | qrcode.js, html5-qrcode or jsQR (for camera-based scanning) |
| **Other Requirements** | Modern browser with camera access permissions enabled |
| **Internet** | Required only for the initial page load (offline after that) |

3.2 HARDWARE REQUIREMENTS

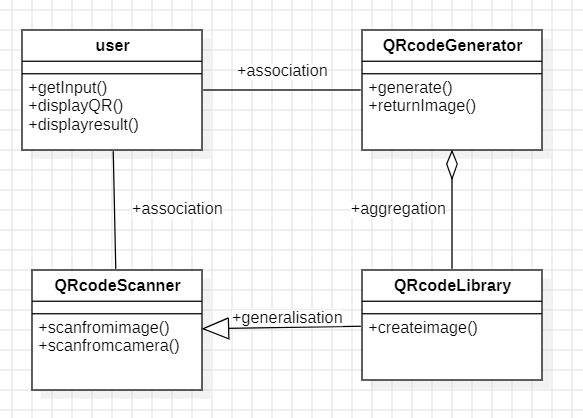
|  |  |
| --- | --- |
| **Component** | **Minimum Specification** |
| **Processor** | Dual-core (Intel/AMD/ARM) or higher |
| **RAM** | 2 GB minimum (4 GB recommended) |
| **Display** | 720p or higher resolution screen |
| **Storage** | Minimal (application runs in browser) |
| **Camera** | Integrated or external camera for QR scanning |
| **Input Devices** | Keyboard and Mouse / Touchscreen (optional) |
| **Device Compatibility** | PC, Laptop, Smartphone, Tablet |

**4. ARCHITECTURE DIAGRAMS**

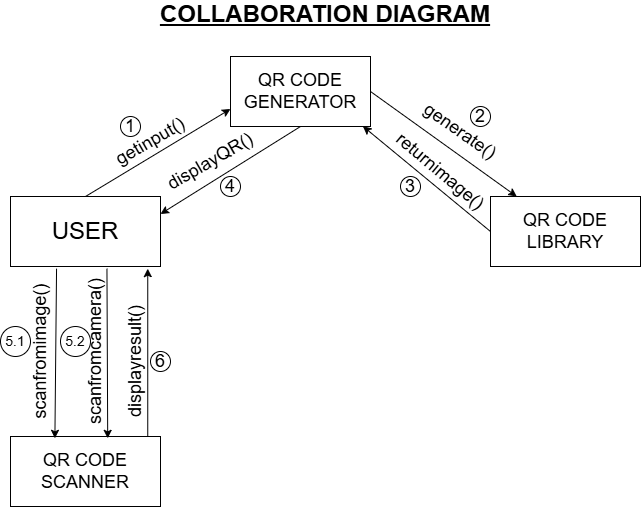
4.1 UML Diagram – 1 **USE CASE DIAGRAM**

****

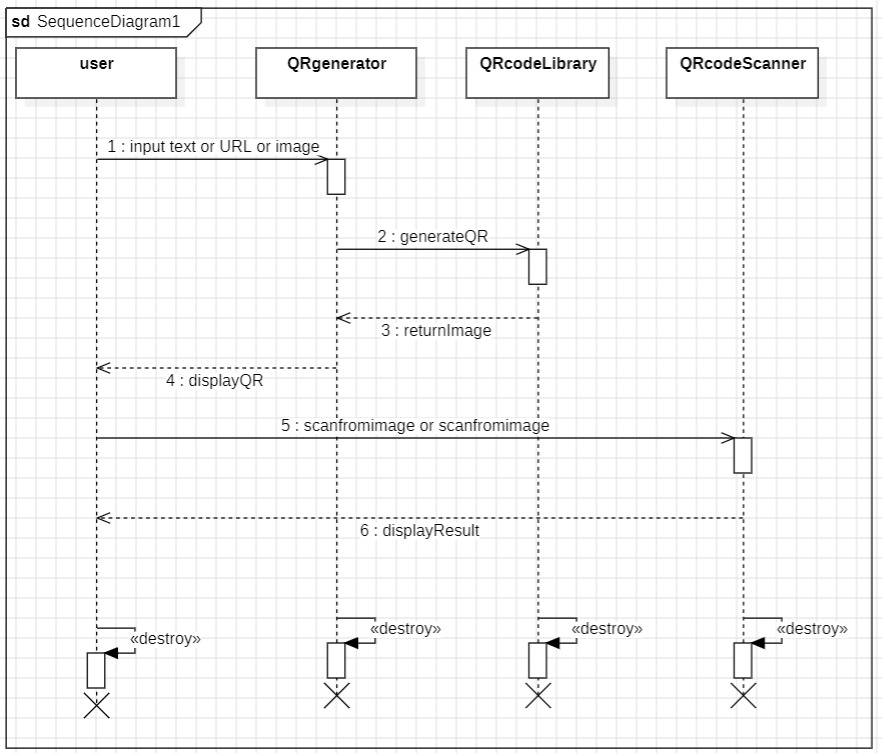
4.2 UML Diagram – 2 **CLASS DIAGRAM**



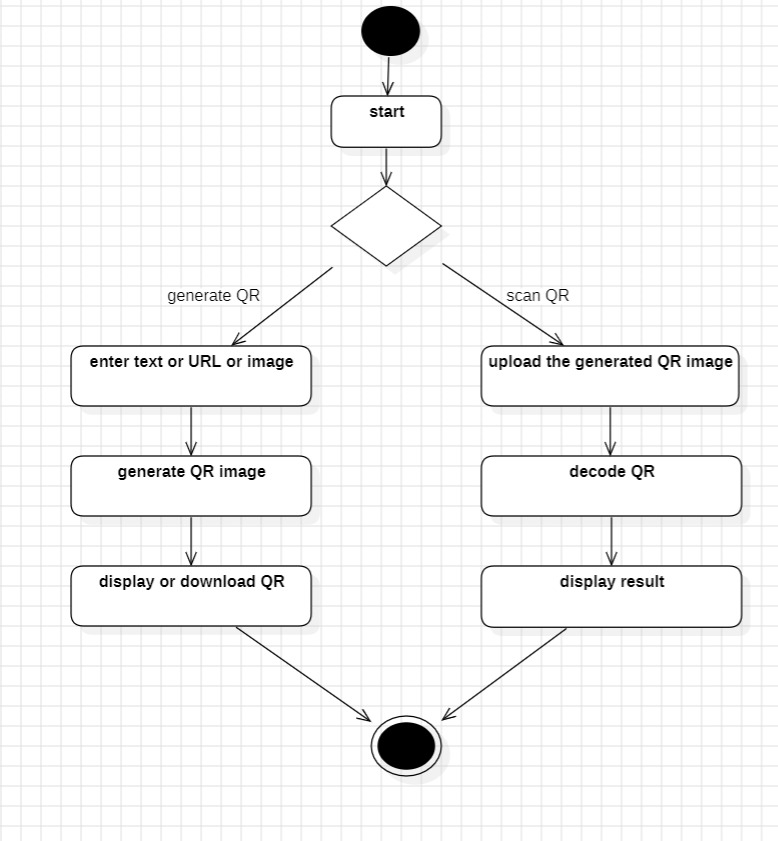
4.3 UML Diagram – 3 **COLLABORATION DIAGRAM**

****

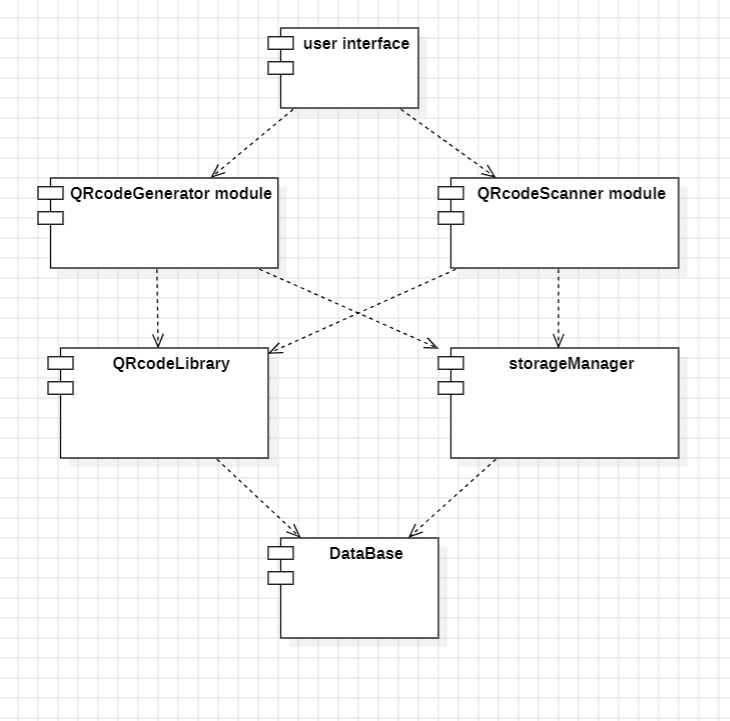
4.4 UML Diagram – 4 **SEQUENTIAL DIAGRAM**

****

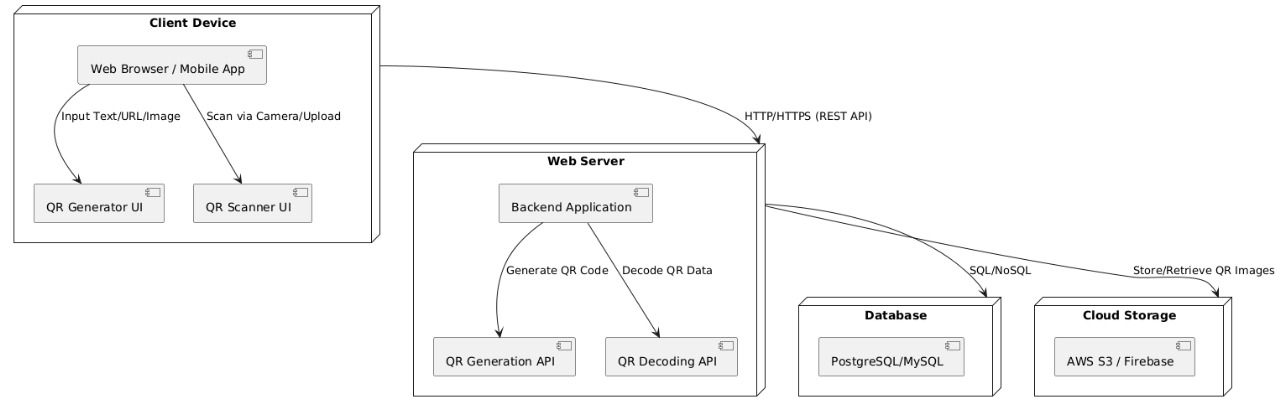
4.5 UML Diagram – 5 **ACTIVITY DIAGRAM**

****

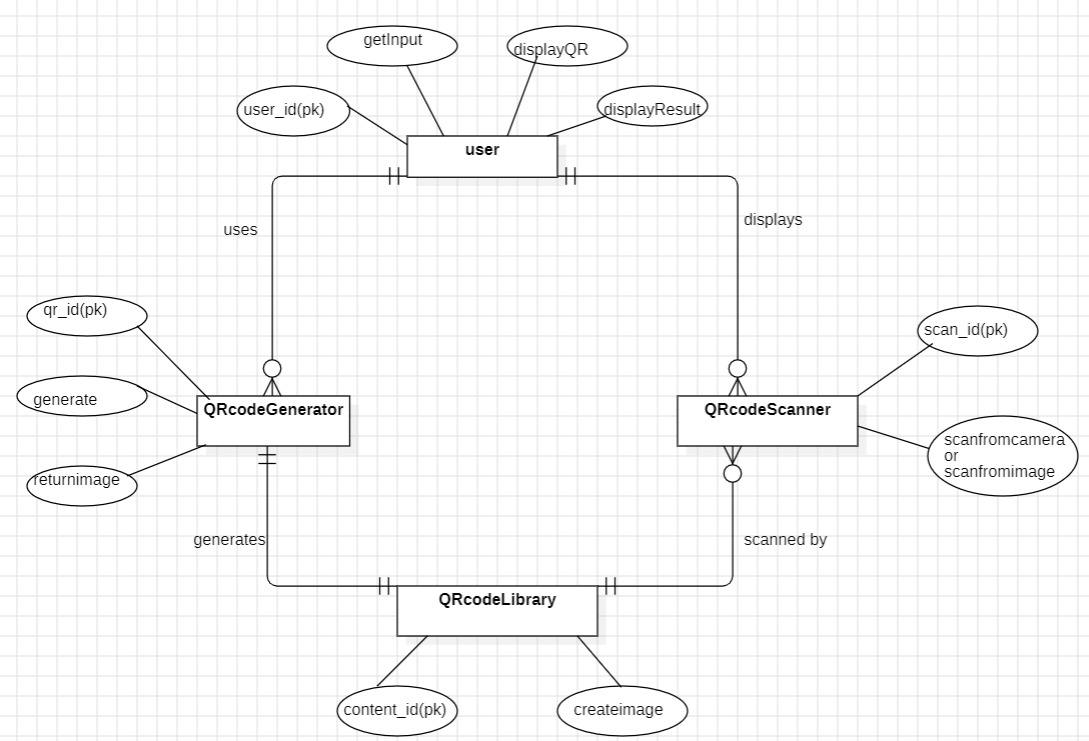
4.6 UML Diagram – 6 **COMPONENT DIAGRAM**

****

4.7 UML Diagram – 7 **DEPLOYMENT DIAGRAM**

****

**4.8 Entity Relationship Diagram**

****

**5. SOFTWARE DESIGN**

The software design of the QR Code Generator and Scanner project follows a modular, client-side architecture that leverages modern web technologies to offer a lightweight, efficient, and privacy-respecting user experience. The application is divided into key functional modules and is built entirely using HTML, CSS, and JavaScript, requiring no server-side logic.

5.1 Architectural Design

* **Application Type:** Single Page Application (SPA)
* **Execution Mode:** Fully client-side (browser-based)
* **Technologies Used:** HTML5, CSS3, JavaScript (Vanilla)
* **Libraries Integrated:**
  + qrious: For generating QR codes
  + jsQR: For decoding QR codes from uploaded images
* **Rendering Engine:** Browser DOM & Canvas API

5.2 Design Modules

1. QR Code Generator Module
   * Accepts text or image file as input
   * Encodes the input as a QR code using the qrious library
   * Displays QR code on an HTML5 canvas
   * Offers download option for QR code image
2. QR Code Scanner Module
   * Accepts uploaded image containing QR code
   * Uses FileReader and Canvas API to extract image data
   * Decodes image data using jsQR library
   * Displays extracted text or renders embedded image (if applicable)
3. User Interface Module
   * Clean, responsive layout using CSS Flexbox and media queries
   * Interactive elements: input fields, buttons, image previews, status messages
   * Accessibility features: keyboard navigation, high contrast, mobile-friendly

5.3 Data Flow Summary

User Input

|

|---> QR Generator Module --> Canvas Output + Download Option

|

|---> QR Scanner Module --> Image Decode --> Output Text or Image

5.4 Layered Design View

* **Presentation Layer:** HTML + CSS for UI rendering
* **Application Logic Layer:** JavaScript functions for QR creation and scanning
* **Data Layer:** In-memory data only (no storage or transmission)

5.5 Design Features

* **Offline Functionality:** Once loaded, runs fully offline
* **Cross-Browser Compatibility:** Tested on Chrome, Firefox, Edge, Safari
* **Cross-Device Responsive Design:** Works on smartphones, tablets, and desktops
* **Security:** All processing happens client-side—no data sent externally

5.6 Future Enhancements Design Road-Map

* **Custom QR Code Styling:** Color options, embedded logos
* **Camera-Based Real-Time Scanning:** Via WebRTC (getUserMedia)
* **Offline QR History:** LocalStorage-based scan history
* **Data Format Detection:** Auto-detect links, contact info, Wi-Fi credentials

This design ensures the tool is scalable, maintainable, and well-structured for future expansion, while remaining performant and user-focused for real-time applications.

**6. MODULE DESCRIPTION**

The project is divided into three core modules, each responsible for a distinct part of the application’s functionality. These modules are designed to operate independently while working seamlessly together to deliver an efficient, lightweight, and user-friendly QR code tool.

6.1 OR Code Generator Module:

**Objective:**  
To convert user-provided input (text, URL, or base64 image data) into a scannable QR code that can be displayed and downloaded.

**Features:**

* Accepts text or image input.
* Encodes input into a QR code using the qrious library.
* Displays the generated QR code using HTML5 <canvas>.
* Provides a download link for users to save the QR code as a PNG file.

**Technologies Used:**  
JavaScript, Canvas API, qrious.js

6.2 QR Code Scanner Module

**Objective:**  
To decode a QR code from an uploaded image and display its embedded content (text or image).

**Features:**

* Accepts QR code image uploads.
* Renders image on an off-screen canvas to extract pixel data.
* Uses the jsQR library to decode QR data from image pixels.
* Displays decoded content and renders embedded images if applicable.

**Technologies Used:**  
JavaScript, Canvas API, FileReader API, jsQR

6.3 User Interface (UI) Module

**Objective:**  
To present an intuitive and responsive interface that supports QR generation, image upload, decoding, and result display.

**Features:**

* Responsive layout that adapts across devices (mobile, desktop).
* Input fields for text and file selection.
* Buttons to trigger generate/scan actions.
* Dynamic messages for user feedback (success, error, instructions).
* Image preview for scanned QR codes.

**Technologies Used:**  
HTML5, CSS3 (Flexbox), JavaScript DOM manipulation

**7. IMPLEMENTATION**

7.1 Source Code: Integrated HTML Page

<!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 and Scanner</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; background: #f4f4f4; text-align: center; }

h1, h2 { color: #333; }

input[type="text"], input[type="file"] { margin: 10px; padding: 8px; width: 80%; max-width: 400px; }

#drop-zone {

margin: 20px auto;

padding: 20px;

border: 2px dashed #0078d4;

background: #eaf4ff;

cursor: pointer;

}

button { margin: 10px; padding: 10px 20px; background: #0078d4; color: #fff; border: none; cursor: pointer; }

button:hover { background: #005ea2; }

canvas, img { margin: 20px auto; background: #fff; box-shadow: 0 0 10px rgba(0,0,0,0.2); }

a { display: inline-block; margin-top: 10px; color: #0078d4; text-decoration: none; }

#scan-image {

width: 100%;

max-width: 100%;

height: auto;

display: none;

border: 1px solid #ccc;

padding: 5px;

}

#warning {

color: red;

margin-top: 10px;

}

#scan-container {

display: flex;

flex-direction: column;

align-items: center;

}

#progress-bar {

width: 80%;

max-width: 400px;

margin: 10px auto;

display: none;

}

</style>

</head>

<body>

<h1>QR Code Generator and Scanner</h1>

<div>

<input type="text" id="text-input" placeholder="Enter text or URL" />

<input type="file" id="image-input" accept="image/\*" />

<div id="drop-zone">Drag and drop an image here</div>

<progress id="progress-bar" value="0" max="100"></progress>

<button onclick="uploadAndGenerateQR()">Generate QR Code</button>

</div>

<canvas id="qr-code" width="250" height="250"></canvas>

<br />

<a id="download-link" style="display: none" download="qr-code.png">Download QR Code</a>

<hr />

<h2>QR Code Scanner (Image Upload)</h2>

<div id="scan-container">

<input type="file" id="qr-scan-input" accept="image/\*" />

<button onclick="decodeQR()">Decode QR Code</button>

<pre id="decoded-output" style="white-space: pre-wrap; text-align: left; max-width: 90%; margin-top: 10px;"></pre>

<div id="warning"></div>

<img id="scan-image" alt="Decoded image will appear here" />

</div>

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

<script src="https://unpkg.com/jsqr/dist/jsQR.js"></script>

<script>

const qrCanvas = document.getElementById('qr-code');

const downloadLink = document.getElementById('download-link');

const qr = new QRious({ element: qrCanvas, size: 250 });

const progressBar = document.getElementById('progress-bar');

const dropZone = document.getElementById('drop-zone');

dropZone.addEventListener('dragover', e => {

e.preventDefault();

dropZone.style.background = '#d0eaff';

});

dropZone.addEventListener('dragleave', () => {

dropZone.style.background = '#eaf4ff';

});

dropZone.addEventListener('drop', e => {

e.preventDefault();

dropZone.style.background = '#eaf4ff';

const files = e.dataTransfer.files;

if (files.length > 0) {

document.getElementById('image-input').files = files;

}

});

function uploadAndGenerateQR() {

const text = document.getElementById('text-input').value.trim();

const file = document.getElementById('image-input').files[0];

if (file) {

if (!navigator.onLine) {

const reader = new FileReader();

reader.onload = () => {

const base64Data = reader.result;

if (base64Data.length > 2950) {

alert("⚠️ Image too large to encode offline. Try using a smaller image or connect to the internet.");

return;

}

qr.value = base64Data;

showDownload();

};

reader.readAsDataURL(file);

} else {

const formData = new FormData();

formData.append('image', file);

const apiKey = 'b094e5930b7b22dd6c36edde6379b02c';

progressBar.style.display = 'block';

const xhr = new XMLHttpRequest();

xhr.open('POST', `https://api.imgbb.com/1/upload?key=${apiKey}`);

xhr.upload.onprogress = e => {

if (e.lengthComputable) {

const percentComplete = (e.loaded / e.total) \* 100;

progressBar.value = percentComplete;

}

};

xhr.onload = () => {

progressBar.style.display = 'none';

const data = JSON.parse(xhr.responseText);

if (data.success) {

qr.value = data.data.url;

showDownload();

} else {

alert("Image upload failed. Try again.");

}

};

xhr.onerror = () => {

progressBar.style.display = 'none';

alert("Upload failed. Check your connection or API key.");

};

xhr.send(formData);

}

} else if (text) {

qr.value = text;

showDownload();

} else {

alert('Please enter text, URL, or choose an image.');

}

}

function showDownload() {

const dataURL = qrCanvas.toDataURL('image/png');

downloadLink.href = dataURL;

downloadLink.style.display = 'inline-block';

const toast = document.createElement('div');

toast.textContent = '✅ Image uploaded and QR code generated!';

toast.style.position = 'fixed';

toast.style.bottom = '20px';

toast.style.right = '20px';

toast.style.background = '#28a745';

toast.style.color = '#fff';

toast.style.padding = '10px 20px';

toast.style.borderRadius = '5px';

toast.style.boxShadow = '0 0 10px rgba(0,0,0,0.2)';

document.body.appendChild(toast);

setTimeout(() => toast.remove(), 3000);

navigator.clipboard.writeText(qr.value).then(() => {

console.log('Copied to clipboard');

});

}

function decodeQR() {

const file = document.getElementById('qr-scan-input').files[0];

if (!file) return;

const img = new Image();

const reader = new FileReader();

reader.onload = () => {

img.onload = () => {

const canvas = document.createElement('canvas');

canvas.width = img.width;

canvas.height = img.height;

const ctx = canvas.getContext('2d');

ctx.drawImage(img, 0, 0);

const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);

const code = jsQR(imageData.data, canvas.width, canvas.height);

const scanImg = document.getElementById('scan-image');

const warning = document.getElementById('warning');

scanImg.style.display = 'none';

warning.textContent = '';

if (code) {

const data = code.data;

const output = document.getElementById('decoded-output');

output.textContent = data;

if (data.startsWith('data:image/')) {

scanImg.src = data;

scanImg.style.display = 'block';

} else if (data.startsWith('http')) {

output.innerHTML += `<br><a href="${data}" target="\_blank">Open Link</a>`;

} else if (data.startsWith('upi:')) {

output.innerHTML += `<br><strong>UPI Link:</strong> <a href="${data}">${data}</a>`;

}

} else {

warning.textContent = 'No QR code found.';

}

};

img.src = reader.result;

};

reader.readAsDataURL(file);

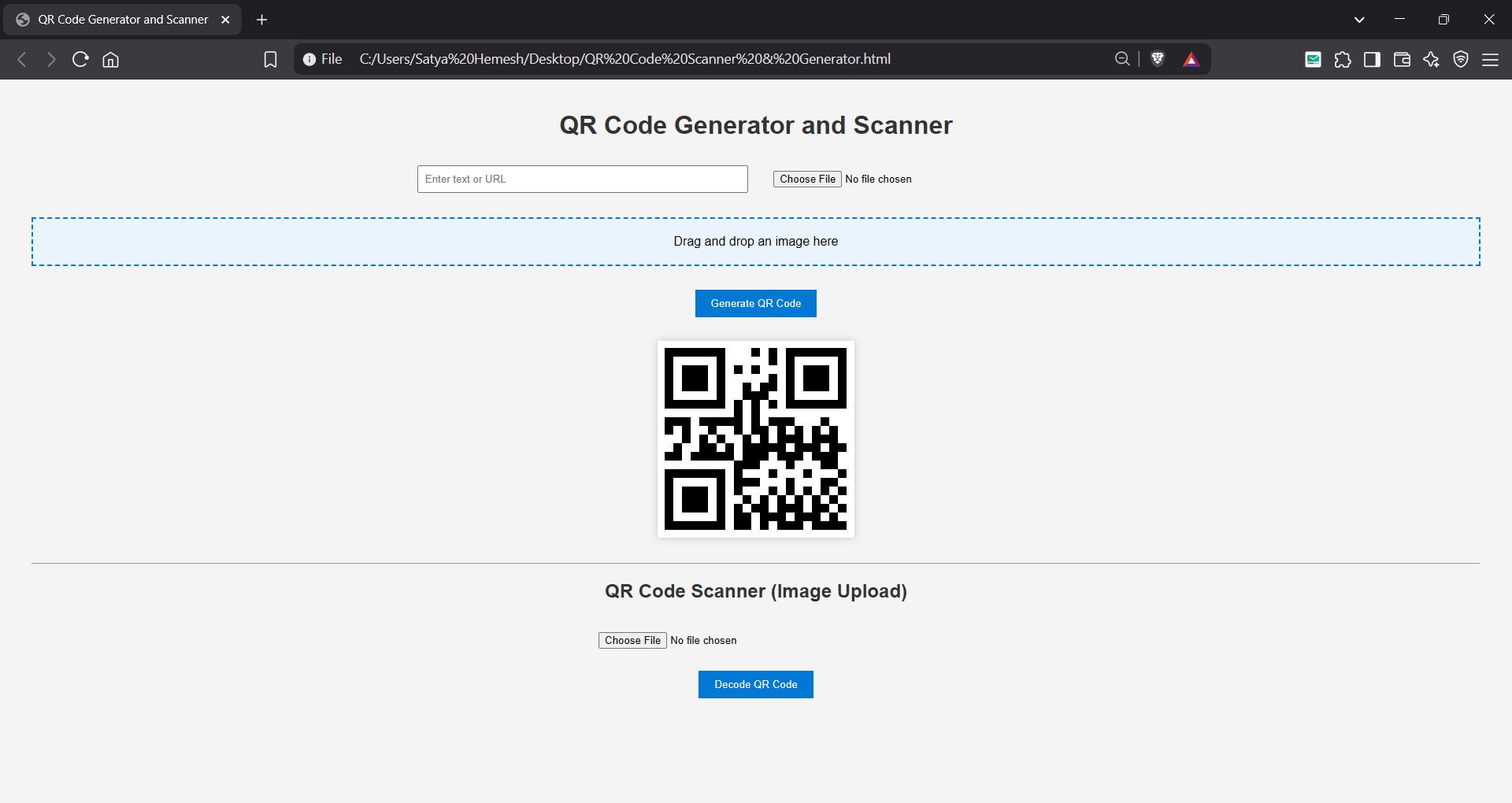
}

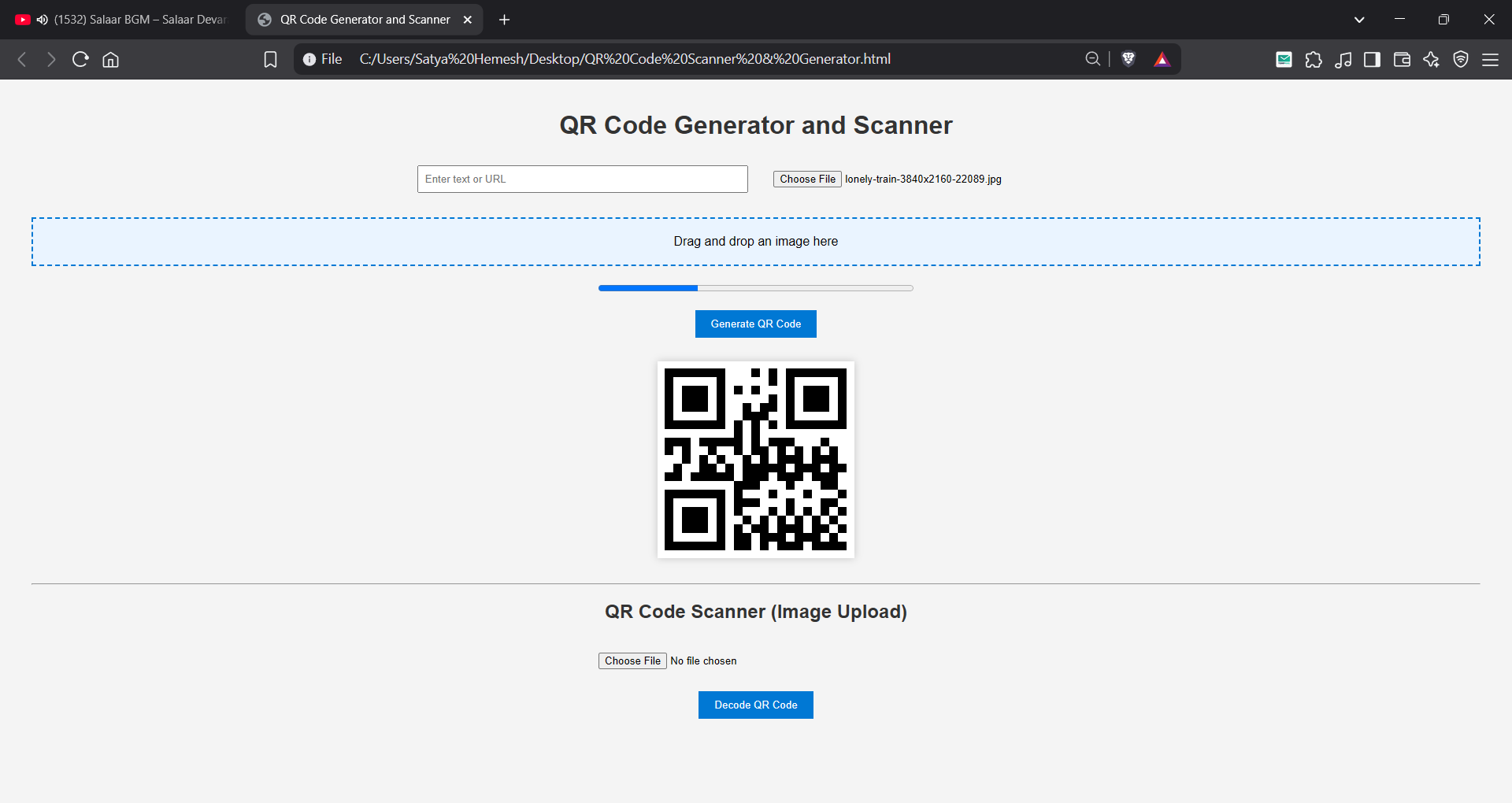
</script>

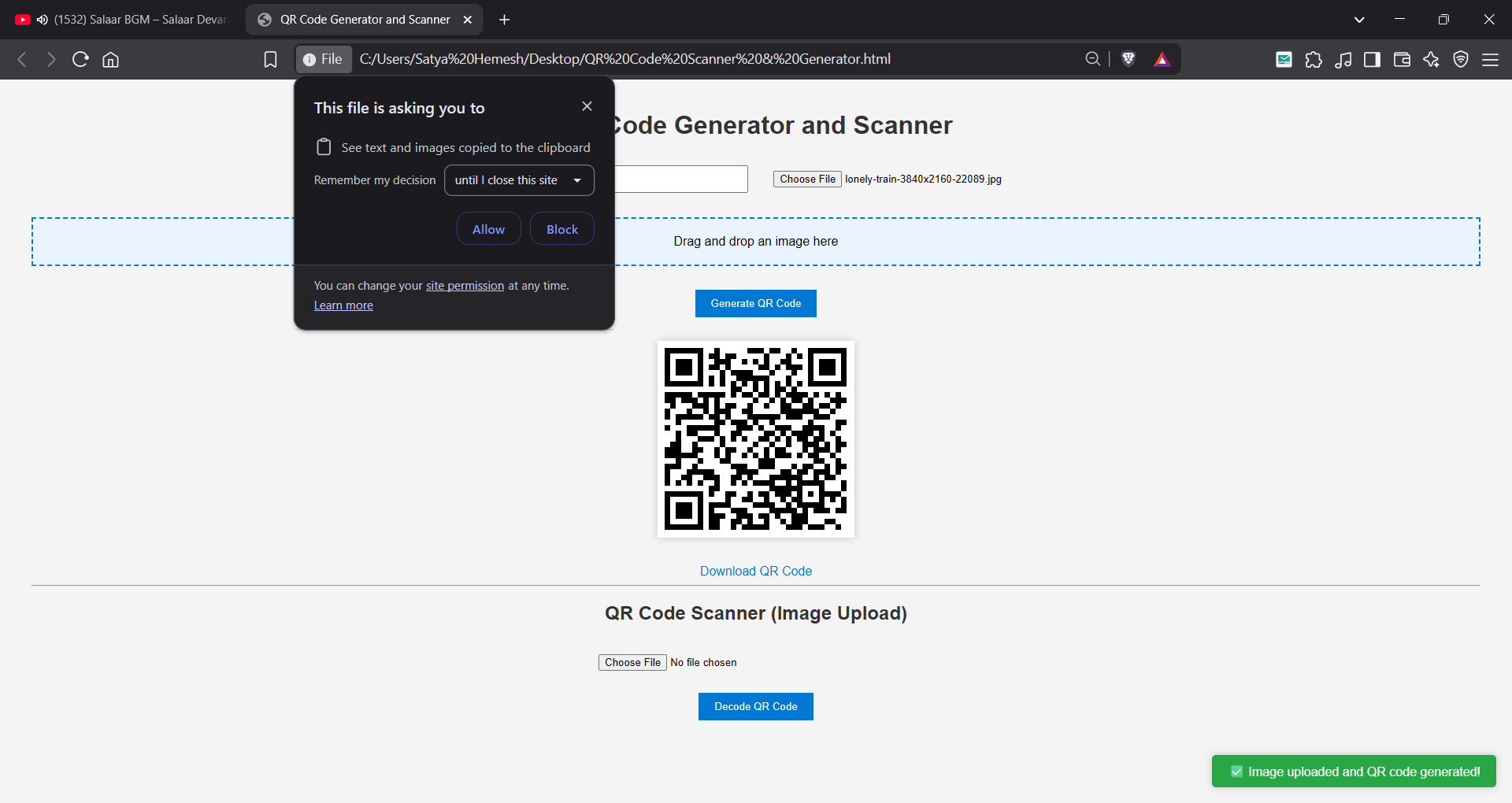
</body>

</html>

**8. OUTPUT SCREENS**

****

****

****

**9. TESTING**

This section outlines the testing performed on the QR Code Generator and Scanner web application to ensure its functional correctness, compatibility, and usability. The application was tested across various scenarios, browsers, and devices to verify consistent behavior.

9.1 Testing Objectives:

* To validate the core functionality of QR code generation from text and image input.
* To ensure accurate decoding of QR codes using uploaded image files.
* To check compatibility across devices and browsers.
* To verify UI responsiveness and error handling.

9.2 Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Result** |
| TC01 | Generate QR from plain text | "Hello World" | QR code is generated and shown in canvas | ✅ Pass |
| TC02 | Generate QR from a URL | "<https://youtube.com>" | QR code generated, downloadable PNG provided | ✅ Pass |
| TC03 | Generate QR from uploaded image (offline) | Small image file | Encoded image as QR code on canvas | ✅ Pass |
| TC04 | Generate QR from uploaded image (online) | Image file (with internet) | Image is uploaded, QR is generated with URL | ✅ Pass |
| TC05 | Decode QR from uploaded QR image | PNG image with embedded text | Decoded text is displayed | ✅ Pass |
| TC06 | Decode QR with no QR present in image | Random image file | Warning: "No QR code found." | ✅ Pass |
| TC07 | Attempt to generate QR with empty input | No input | Alert: "Please enter text, URL, or choose image." | ✅ Pass |
| TC08 | Upload large image while offline | >3000 byte image | Alert: "Image too large to encode offline" | ✅ Pass |
| TC09 | Click download link after generating QR | QR code visible | PNG file downloads successfully | ✅ Pass |
| TC10 | Drag & drop image to input | Drag a valid image file | Image is accepted and ready for generation | ✅ Pass |

9.3 Compatibility Testing:

The application was tested on multiple browsers and operating systems:

|  |  |  |
| --- | --- | --- |
| **Browser** | **OS/Device** | **Status** |
| Chrome (latest) | Windows 10 Laptop | ✅ Pass |
| Firefox (latest) | Ubuntu Linux | ✅ Pass |
| Edge (Chromium) | Windows 11 | ✅ Pass |
| Safari | iPhone 13 (iOS 17) | ✅ Pass |
| Chrome | Android Phone | ✅ Pass |

9.4 Responsive UI Testing:

|  |  |  |
| --- | --- | --- |
| **Device** | **Test** | **Result** |
| Desktop (1080p) | Layout aligned, buttons visible | ✅ Pass |
| Tablet | Responsive input and QR section | ✅ Pass |
| Mobile (Portrait) | Buttons and inputs usable | ✅ Pass |

9.5 Error Handling:

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Behaviour** | **Result** |
| No input given | Alert shown | ✅ Pass |
| Invalid/large image file (offline) | Alert: image too large to encode | ✅ Pass |
| QR decode fails | Message: "No QR code found." | ✅ Pass |
| Server or API fails (online upload) | Alert: Upload failed | ✅ Pass |

9.6 Performance Testing:

* **Load Time:** Instant on reload after first use (thanks to browser caching).
* **Offline Support:** Works seamlessly once fully loaded, including generation and decoding from uploaded files.
* **CPU/RAM Usage:** Low, suitable for low-end mobile devices.

9.7 Usability and Accessibility

* Input elements are clear and intuitive.
* Button feedback and alert messages provide real-time guidance.
* Text size and contrast are readable on all tested devices.

**10. CONCLUSION**

The QR Code Generator and Scanner project is adequately shown to create a light, effective, and purely client-side web application that can create as well as scan QR codes. With the use of contemporary web technologies like HTML, CSS, and JavaScript, the project accomplishes its mission of providing a smooth, safe, and user-friendly method for live data encoding and reading without any dependency on third-party apps or server-side servers.

The app offers several input modes (text, URL, and images), accommodates image file upload and drag-and-drop, and offers real-time QR code decoding using uploaded images. Its responsive nature makes it compatible with desktops, tablets, and mobile devices, making it extremely accessible. In addition, offline capabilities and client-side computation improve usability and privacy, particularly in low-connectivity settings.

By thoroughly testing and implementing the features, this project confirms the viability of employing browser-native APIs and open-source libraries such as qrious and jsQR in providing a working, real-time QR solution. The system overall is a useful and scalable tool with possible use cases in education, marketing, healthcare, and personal, everyday life.

**11. FUTURE ENHANCEMENTS**

While the current version of the QR Code Generator and Scanner meets the essential requirements of generating and decoding QR codes, several enhancements can be implemented to further improve the tool’s capability, usability, and performance. Potential future upgrades include:

11.1 Real-Time Camera-Based Scanning

* Integrate real-time scanning directly from the device’s camera using WebRTC.
* Allow continuous scanning without needing image uploads.

11.2 Custom QR Code Styling

* Allow users to change QR code color, size, and error correction levels.
* Add support for embedding logos or profile images into the QR code.
* Provide design templates for professional/business use.

11.3 Scan History (Local Storage)

* Store previously scanned QR results in browser local storage.
* Display a history panel with timestamps for quick access.

11.4 QR Code Format Support

* Support additional data types:
  + Wi-Fi credentials
  + vCard/contact information
  + Geo locations
  + UPI/payment codes
  + Event calendar links

11.5 Accessibility Features

* Add voice guidance for visually impaired users.
* Enable keyboard navigation and ARIA labels for screen readers.

11.6 Multilingual Support

* Provide language options for a broader user base.
* Use internationalization (i18n) techniques for text labels and messages.

11.7 Progressive Web App (PWA)

* Convert the tool into a PWA to support full offline use with installable features.
* Allow users to use the tool as a standalone mobile app.

11.8 Backend Integration:

* Add a server backend to store QR codes, manage user accounts, or allow cloud-based access (optional for enterprise use).
* Enable QR code expiration or tracking analytics for business purposes.

11.9 Error Correction Customization:

* Let users adjust the QR error correction level (L, M, Q, H) for balancing design and reliability.

11.10 Security Enhancements:

* Scan QR code content for potential phishing or malicious links.
* Display security warnings or block known unsafe URLs.

**12. REFERENCES**

1. qrious.js – QR Code Generator Library

URL: <https://github.com/neocotic/qrious>

2.jsQR – Pure JavaScript QR Code Scanner

URL: <https://github.com/cozmo/jsQR>

3. imgbb – Free Image Hosting API

URL: <https://api.imgbb.com/>

4. MDN Web Docs – HTML, CSS, and JavaScript Documentation

URL: <https://developer.mozilla.org/>

5. Web APIs: FileReader, Canvas, MediaDevices, Blob, and URL

URL: <https://developer.mozilla.org/en-US/docs/Web/API>

6. W3Schools – Web Technology Tutorials and Examples

URL: <https://www.w3schools.com/>

7. Stack Overflow – Developer Community Support

URL: <https://stackoverflow.com/>

8. Google Chrome DevTools – Web Development Debugging Tools

URL: <https://developer.chrome.com/docs/devtools/>

9. jsDelivr CDN – Hosting JavaScript Libraries

URL: <https://www.jsdelivr.com/>

10. GitHub – Open Source Contributions and Library Hosting

URL: <https://github.com/>