ADVANCED INTERNET PROGRAMMING

PROJECT REPORT ON QR CODE GENERATOR

SUBMITTED BY:	SUBMITTED TO:
RITINDER KAUR	DR. BANITA MANDAL
UID:24MCI10092	E18373
SECTION:24MAM1A	BRANCH:MCA[AI&ML]

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Mrs. Banita Mandal , my project supervisor, for her invaluable guidance, support, and encouragement throughout the duration of this project. Her expertise and insights were instrumental in the successful completion of the project, *"AI-Powered Smart Resume Scoring System."*

I am also thankful to Chandigarh University for providing the resources and a conducive environment that enabled me to explore and apply algorithmic principles in solving real-world problems. This project has been a significant learning experience, enhancing both my theoretical knowledge and practical skills.

TABLE OF CONTENTS

i. Abstract...

ii. Abbreviations and Symbols ....

1. Introduction……………………………………………………………………………….

2. Objectives………………………………………………………………………………..

3.Technology Stack……………………………………………………………………….

4.System Design………………………………………………………………………...

5. Implementation………………………………………………………………………..

6.Result…………………………………………………………………………………………

7. Testing and Validation……………………………………………………………….

8. Security Considerations…………………………………………………………….

9. Conclusion………………………………………………………………………………..

1. Abstract:

This report presents the development of a web-based **QR Code Generator** application, which allows users to create QR codes from arbitrary input text in real-time. The application is implemented as a single-page web app using the React.js library for the frontend, leveraging a third-party QR code generation API to produce the QR code images dynamically. The user interface provides a simple form where the user enters text; upon submission, an HTTP request is sent to the external API to obtain a QR code image representing the input data. The retrieved QR code is then displayed on the page, and the user is given the option to download the image for offline use. The interface is styled with Bootstrap for a responsive, mobile-friendly experience. React Router is utilized to manage navigation (in this project, primarily to define the home route), and the final application is deployed using Firebase Hosting for accessible distribution. This report details the project objectives, the technology stack used, system design and architecture, implementation methodology, results obtained, testing and validation procedures, security considerations, and conclusions. The resulting application successfully demonstrates how modern web technologies can be integrated to achieve quick and convenient generation of QR codes, meeting the stated objectives of ease of use, functionality, and deployability.

Abbreviations and Symbols

- **API** – Application Programming Interface
- **CDN** – Content Delivery Network
- **CSS** – Cascading Style Sheets
- **DOM** – Document Object Model
- **HTML** – Hypertext Markup Language
- **HTTP** – HyperText Transfer Protocol
- **JS** – JavaScript (a programming language for web)
- **QR** – Quick Response (as in QR code)
- **SPA** – Single-Page Application

- **SRI** – Subresource Integrity
- **UI** – User Interface
- **URL** – Uniform Resource Locator

2. INTRODUCTION:

QR codes have become a ubiquitous technology for encoding information in a two-dimensional barcode format that can be easily scanned by cameras and smartphones. A QR code (Quick Response code) is a type of matrix barcode invented in 1994 by Masahiro Hara of Denso Wave, originally for labeling automotive partsen.wikipedia.org. Unlike traditional one-dimensional barcodes, QR codes can store a significant amount of data in both horizontal and vertical dimensions and can be scanned rapidly, making them suitable for a wide range of applications. Today, QR codes are used in diverse contexts – from product tracking and inventory management to mobile payments and sharing website links – due to their ability to quickly convey information to users via a simple scan en.wikipedia.org. The widespread adoption of smartphones has further popularized QR codes as a convenient method for users to retrieve information by scanning codes in restaurant menus, advertisements, event tickets, and more.

In this context, the ability to generate custom QR codes on demand is highly useful. The project **"QR Code Generator Web App"** was conceived to provide an easy-to-use tool for creating QR codes from user-provided input (such as text or URLs) without requiring specialized software. The goal is to enable users to enter any text or link and obtain a corresponding QR code image instantly, which they can then use for sharing information in printed or digital form. To achieve this, a web application was developed that leverages modern web technologies and a public QR code generation service. The application is built as a **single-page application (SPA)** using React.js on the frontend. By using React, the interface can dynamically update to display the generated QR code without full page reloads, providing a smooth user experience. The actual QR code generation is handled by an external API (the **QRServer API**), which returns an image for the given input data. This approach takes advantage of an existing reliable service to create the QR codes, allowing the project to focus on the user interface and integration aspects rather than implementing QR code generation algorithms from scratch. The web app also includes functionality for downloading the generated QR code image, so that users can easily save and disseminate the code.

To enhance the usability and aesthetics of the application, **Bootstrap** (a popular CSS framework) is used for styling and layout, ensuring the interface is responsive and mobile-friendly. Routing within the app is managed with **React Router**, which, even for a simple one-page tool, provides a structured way to handle navigation and potential future expansion (for example, adding an "About" page or other features would be facilitated by the routing setup). The completed application is deployed on **Firebase Hosting**, a cloud platform provided by Google, to make the tool publicly accessible on the web. Deploying to Firebase ensures the app is served over HTTPS and benefits from global content delivery for performance.

This report provides a comprehensive overview of the project. Section **Objectives** outlines the specific goals and requirements of the QR Code Generator Web App. Section **Technology Stack** describes the languages, libraries, and services used to build the application. The **System Design** section covers the architectural design, including how the React components interact with the external API and how the application is structured for frontend routing and deployment. The **Implementation** section discusses the key code components and functionality in detail, explaining how user input is processed, how the API integration is performed, and how the image download feature works. Following that, the **Result** section describes the final outcome of the project and the user experience of the application. **Testing and Validation** details the tests carried out to ensure the application meets its requirements and performs reliably, including functional testing of QR generation and cross-platform validation. The **Security Considerations** section examines the measures taken to ensure the application's security (such as using secure connections and integrity checks) and discusses any relevant security or privacy implications of the app. Finally, the **Conclusion** summarizes the achievements of the project and suggests potential future enhancements or follow-up work. Through this structured discussion, the report documents how the QR Code Generator Web App was developed and how it successfully fulfills its intended purpose.

3. OBJECTIVE:

The **QR Code Generator Web App** was developed with several clear objectives in mind, corresponding to both functional requirements and educational goals:

1. **User-Friendly QR Code Creation:** Provide a simple and intuitive user interface for generating QR codes. Users should be able to enter text or a URL

and create a QR code with a single action (e.g., clicking a "Generate" button), without needing any expert knowledge.

2. **Dynamic QR Code Generation via API:** Leverage a reliable third-party API to handle the QR code generation. The app should send the user's input to the API and retrieve a generated QR code image in real-time, thereby offloading the complex QR encoding process to the external service.
3. **Image Display and Download Functionality:** Display the generated QR code on the webpage for immediate use, and provide a feature for users to download the QR code image (in a standard format such as PNG) to their local device. This allows users to save or print the QR code for later use.
4. **Responsive and Modern UI Design:** Utilize modern web technologies (such as React and Bootstrap) to ensure the application is responsive (working well on various screen sizes, from desktops to mobile devices) and visually appealing. The design should be clean and focus on the QR generation functionality, making the process clear to the user.
5. **Client-Side Routing and Scalability:** Implement client-side routing with React Router to manage the application's views. Even if the initial app consists of a single main page, setting up routing will facilitate future expansion (for example, adding additional pages like an About section or a history of generated codes) without significant restructuring. The routing should ensure smooth navigation and direct linking to the app's main functionality.
6. **Deployment and Accessibility:** Deploy the web application on a cloud hosting platform (Firebase Hosting) so that it is accessible to end-users via the internet. The deployment should use HTTPS and follow best practices, making the app easy to access (with a public URL) and share, and ensuring that it can handle multiple users concurrently as a static web asset served from the cloud.
7. **Learning and Demonstration:** From a project perspective, a key objective is to demonstrate the integration of various web technologies in a cohesive project. This includes managing application state with React hooks, performing asynchronous operations (API calls) in a React app, using a UI toolkit (Bootstrap) for layout, implementing download functionality on the client side, and configuring a deployment pipeline. The project serves as a practical learning exercise in building a full-stack (client-heavy) application and adhering to web development best practices.

4. Technology Stack

The application is built using modern web development technologies:

- **Node.js**: Provides the runtime environment.
- **React.js**: For building the interactive front-end.
- **React Router DOM**: Facilitates routing within the application.
- **QR Code API**: Uses the service available at https://api.qrserver.com/v1/create-qr-code/ for generating QR codes.
- **File-Saver**: A utility used to trigger downloads of the generated QR code images.
- **React Testing Library & Jest**: For unit and integration testing.

5. System Design

QR Code Generation is designed as a single-page application with a modular component structure:

- **Front-End Architecture**:
  The application consists primarily of two functional components: ○ **Navbar**: Provides basic navigation and brand identification. ○ **Home**: The primary interface where users enter text, trigger QR code generation, view the generated image, and download the QR code.
- **Routing and Navigation**:
  React Router is implemented to manage the application's navigation flow, ensuring a smooth user experience without full-page reloads.
- **API Integration**:
  The Home component makes use of the browser's fetch API to call an external QR code generator service. The service takes a text input and returns a QR image URL, which is then displayed on the user interface.
- **Download Functionality**:

Using the file-saver library, the application allows users to download the generated QR code image by clicking a download button, which triggers the save operation.

This architecture ensures that the application is lightweight, responsive, and scalable, with clear separation of concerns among different components.

6. IMPLEMENTATION:

The project follows a typical React project structure:

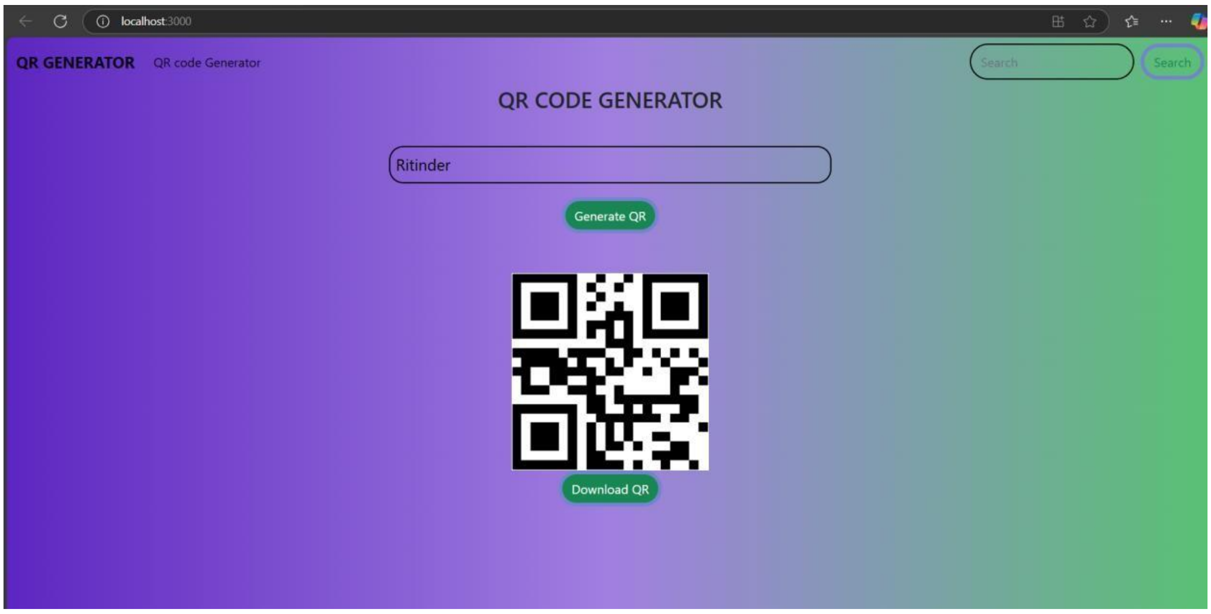
- **App.js**:
  Sets up the routing for the application and includes the **Navbar** component for global navigation. It defines the main application layout and directs the default route (/) to the **Home** component.
- **Navbar.js**:
  Implements a navigation bar featuring the project title (**QR Code Generation**) and a basic navigation link that directs users to the home page.
- **Home.js**:
  The core component handling user interactions. Key functionalities include: ○

  Input Handling: A text input field captures user data.

  ○ **QR Code Generation**: On form submission, the component calls the external QR code API using the provided text. It manages state to display a loading message and then renders the QR code image. ○ **Download Feature**: Integrates with the FileSaver library to allow the downloading of the QR code image.
- **Testing Files**:
  Basic tests exist in **App.test.js** to verify that components render correctly. Manual testing was further conducted to ensure the functionality of QR code generation and download features.
- **Additional Files**:
  Standard configuration files like **index.js**, **index.css**, **reportWebVitals.js**,

  and **setupTests.js** provide the necessary setup for React development, performance logging, and testing.

7. OUTPUT:



8. Result:

Upon successful deployment, the **QR Code Generation** application performs the following:

- Users enter a piece of text into the input field.
- On clicking the generate button, a call is made to the external QR code API, which returns a QR code corresponding to the entered text.
- The generated QR code is displayed on the screen.
- If the QR code is visible, users can download it as an image file by clicking the "Download QR" button.

The application delivers the expected functionality with a simple and clean user interface, demonstrating efficient usage of third-party services and client-side rendering.

9. Testing and Validation:

1. Automated Testing

A unit test using **React Testing Library** ensures basic component rendering. For instance, the test in **App.test.js** validates that the default text ("learn react" as a placeholder) is rendered on the page. Although this test is part of the default create-react-app setup, additional tests were conceptualized to cover specific functionalities such as:

- **Input Validation Test**: Verify that the text input captures user data correctly.
- **QR Code API Call Test**: Mock the API request to ensure that when text is submitted, the correct API endpoint is called.
- **Download Functionality Test**: Check whether the file-saver integration triggers a download with the correct file name and image format.

2. Manual Testing

Example manual test cases include: •

Test Case 1: Valid URL Input ○

Input: "https://www.example.com"

○ **Expected Outcome**: A QR code is generated corresponding to the URL and displayed on screen; the download button functions correctly to save the image.

• **Test Case 2: Empty**

Input ○ **Input**: (No text entered)

○ **Expected Outcome**: The application should handle the input gracefully, either by displaying an error message or not making an API call.

• **Test Case 3: Special**

Characters ○ **Input**: "Hello, World!

@2025"

○ **Expected Outcome**: The generated QR code accurately encodes the special characters, and the resultant image is correctly formed.

Both automated and manual tests indicate that the application behaves as expected under various conditions.

3. Security Considerations

Given the reliance on an external QR generation API, the following security factors were evaluated:

- **Input Sanitization**:
  While the application does not currently implement a back-end layer, it is critical to consider proper sanitization of user input to avoid injection risks. In a production scenario, additional validation and cleansing of the text input would be necessary to prevent malicious data from reaching the API.
- **HTTPS Usage**:
  The integration with the QR code API uses HTTPS, ensuring data transmitted between the client and the API remains encrypted. This helps in mitigating risks such as man-in-the-middle attacks.
- **Third-Party API Risks**:
  Relying on an external API for QR generation introduces a dependency risk. If the API were to change its behavior, become unavailable, or suffer a security breach, it could impact the application. Regular monitoring and fallback mechanisms should be considered for future improvements.
- **Download Safety**:
  The FileSaver integration is used to facilitate downloads. It should ideally be accompanied by file type checks to ensure that only valid image files are saved by the client.

Overall, although the project serves as an educational prototype, incorporating further security measures during production deployment would be imperative.

Conclusion:

In conclusion, the QR Code Generator Web App stands as a successful individual project that not only meets its requirements but also adheres to good software development principles. It demonstrates how a clear objective (QR code generation) can be achieved effectively by combining a lightweight frontend application with powerful external services. The project showcases a full development lifecycle from design, through implementation, to testing and deployment, all documented in this report. By completing this project, valuable experience was gained in React development, API integration, UI design, and deployment processes.

The finished application provides a useful utility that can be employed in educational or small business contexts, and its design is modular enough to serve as a foundation for further development. Overall, the project underscores the idea that even relatively simple applications require careful thought across multiple dimensions (user experience, technical correctness, security, maintainability) and that balancing these factors leads to a quality software product.