



# **HALL TICKET GENERATOR USING QR CODE**

**MAJOR PROJECT GUIDE:**

**PROF. RUPA SAHA**

**DEPARTMENT OF COMPUTER APPLICATION**

**GROUP MEMBERS**

NAME	UNIVERSITY ROLL NO.
SOUMO DHALI	434122010014
IVYLATA DAS	434122020017

---

**NARULA INSTITUTE OF TECHNOLOGY**

81,NILGUNJ ROAD AGARPARA KOLKATA-700109

---

PHONE NO.- 033-25637777/8888

WEBSITE-[www.nit.ac.in](http://www.nit.ac.in)

# CERTIFICATION OF APPROVAL

This is to certify that the following students of MCA 2022-24 Semester IV has satisfactory completed the project titled – “**HALL TICKET GENERATOR USING QR CODE**”. This project is being submitted in partial fulfillment for the award of the degree of Master of Computer Application under **Narula Institute of Technology**.

NAME	UNIVERSITY ROLL NO.
SOUMO DHALI	434122010014
IVYLATA DAS	434122020017

---

Project mentor

**Prof. Rupa Saha**

(Dept. of Computer Application)

**NARULA INSTITUTE OF TECHNOLOGY**

# **CERTIFICATE OF ORIGINALITY**

I and my group has developed this project in 2nd year, 4<sup>th</sup> semester of Master in Computer Application (Major Project). While developing the project no unfair means or illegal copies of software, etc, have been used neither any part of this project nor any documentation have been submitted elsewhere or copied.

SOUMO DHALI

434122010014

IVYLATA DAS

434122020017

# ACKNOWLEDGEMENT

First and foremost, we would like to express our gratitude to our Mentor, **PROF. RUPA SAHA**, Dept. of Computer Application, **NARULA INSTITUTE OF TECHNOLOGY, KOLKATA**, who was a continual source of inspiration. She pushed us to think imaginatively and urged us to do this project without hesitation. Her vast knowledge, extensive experience and professional competence enabled us to successfully accomplish this project.

We have a special thanks to **PROF. DEBRUPA PAUL**, **PROF. SUBHASREE BHATTACHARJEE**, and other Faculties of CA department. This endeavor would not have been possible without their help and supervision. This initiative would not have been a success without the contributions of each and every individual.

## Signature of Group Members

SOUMO DHALI

434122010014

IVYLATA DAS

434122020017

# TABLE OF CONTENT

- Introduction
- Overview
- Scope of the project
- System Requirements
- System Requirement Specification (SRS)
- E-R Diagram
- DFD
- Context Diagram
- Coding
- Testing
- Bibliography
- Conclusion

# **Introduction:**

The Hall Ticket Generator with QR Code Integration is a project aimed at revolutionizing the way hall tickets are issued and managed for various examinations. The traditional method of issuing paper-based hall tickets is not only cumbersome but also prone to errors and misuse. By integrating QR codes into hall tickets, this project seeks to enhance security, reduce the risk of fraud, and improve the overall efficiency of the examination process.

QR codes (Quick Response codes) are two-dimensional bar codes that can store a large amount of information, including student details, examination schedule, and venue details. By scanning the QR code using a smartphone or a dedicated QR code scanner, exam invigilators can quickly verify the authenticity of the hall ticket and access relevant information about the student and the examination.

This project will involve the development of a web-based application that allows examination authorities to generate and issue hall tickets with unique QR codes for each student. The application will also include features for managing examination schedules, seating arrangements, and generating reports.

Overall, the Hall Ticket Generator using QR Code Integration project aims to streamline the process of issuing and managing hall tickets, improve security and efficiency, and enhance the overall examination experience for students and examiners alike.

## **Overview:**

The Hall Ticket Generator with QR Code Integration project modernizes the issuance and management of hall tickets for examinations. Using QR codes, the project enhances security, reduces fraud, and improves efficiency. The web-based application allows authorities to generate hall tickets with unique QR codes for each student, manage schedules, seating arrangements, and generate reports. Overall, the project aims to streamline examinations, improve security, and enhance the examination experience.

## Scope of the Project:

The Hall Ticket Generator with QR Code Integration application offers a wide range of benefits and opportunities for improvement in the examination process. Some of the key scopes of this application include:

**Enhanced Security:** By integrating QR codes into hall tickets, the application ensures that only authentic hall tickets are accepted, reducing the risk of fraud and unauthorized entry.

**Efficiency:** The application streamlines the process of issuing and managing hall tickets, saving time and effort for examination authorities.

**Cost-Effectiveness :** By reducing the need for paper-based hall tickets and manual verification processes, the application helps in saving costs for examination authorities.

Overall, the Hall Ticket Generator with QR Code Integration application has the potential to significantly improve the efficiency, security, and overall experience of the examination process



# **System Requirements:**

## **HARDWARE REQUIREMENTS:**

- System : Pentium i3 Processor.
- Hard Disk : 500 GB.
- Monitor : 15'' LED.
- Input Devices : Keyboard, Mouse.
- Ram : 4 GB.

## **SOFTWARE REQUIREMENTS:**

- Operating system : Windows 10/11.
- Coding Language : JavaScript.
- Frontend : JSP, HTML, CSS, JavaScript, React.
- IDE Tool : Apache Netbeans IDE 16.
- Database : <https://appwrite.io/>.

# **System Requirement Specification(SRS)**

The Hall Ticket Generator using QR Code system will be a stand-alone application that interacts with users through a web-based interface. It will integrate with a database to store hall ticket and examination information.

## ***Product Functions***

- Generate unique QR codes for each hall ticket.
- Store and manage examination schedules and seating arrangements.

## ***User Classes and Characteristics***

- Exam authorities : responsible for generating and managing hall tickets.
- Students : access their hall tickets using QR codes.

## ***Operating Environment***

- The system will be deployed on a web server and accessed through a web browser. It will be compatible with modern web browsers such as Chrome, Firefox, and Safari.
- The system shall allow exam authorities to login and access the hall ticket generation functionality.
- The system shall allow exam authorities to input student details and generate a unique QR code for each hall ticket.

## ***Specific Requirements***

### **Functional Requirements**

- The system shall allow students to access their hall tickets using the QR code.

### **Non-Functional Requirements**

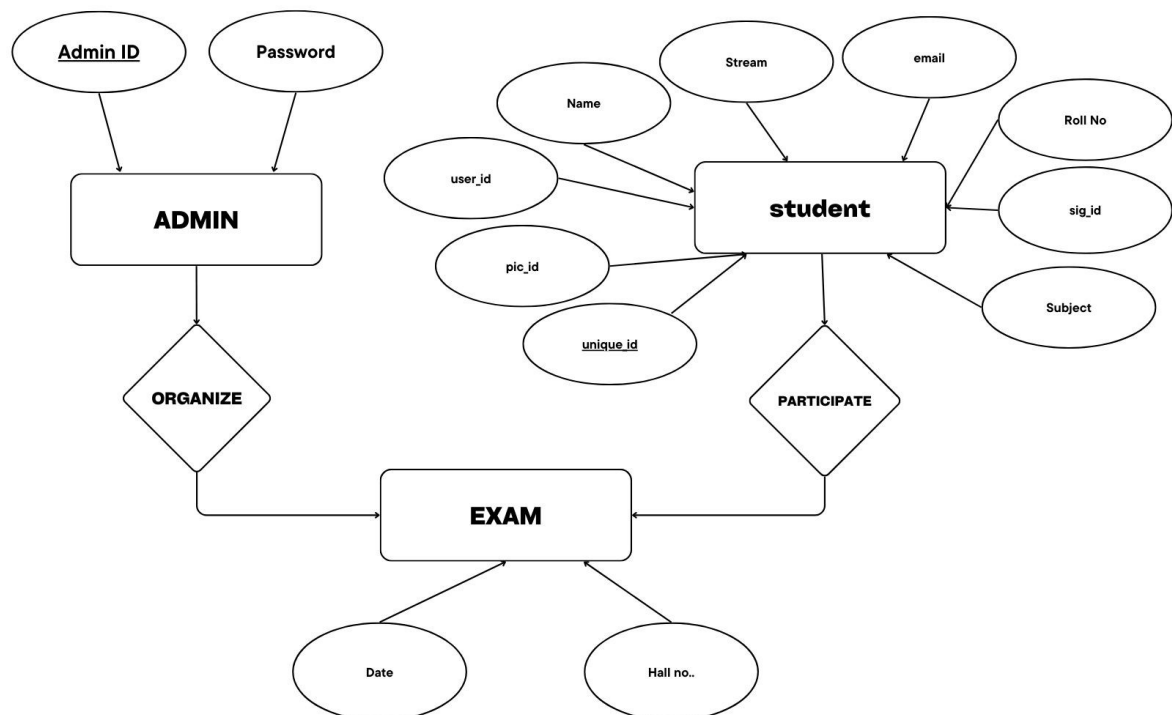
- The system shall be scalable, capable of handling a large number of hall tickets and users.
- The system shall be user-friendly, with an intuitive interface for exam authorities and students.

### **Performance Requirements**

- The system shall generate QR codes quickly and efficiently, with minimal delay.
- The system shall be responsive, providing a seamless user experience.

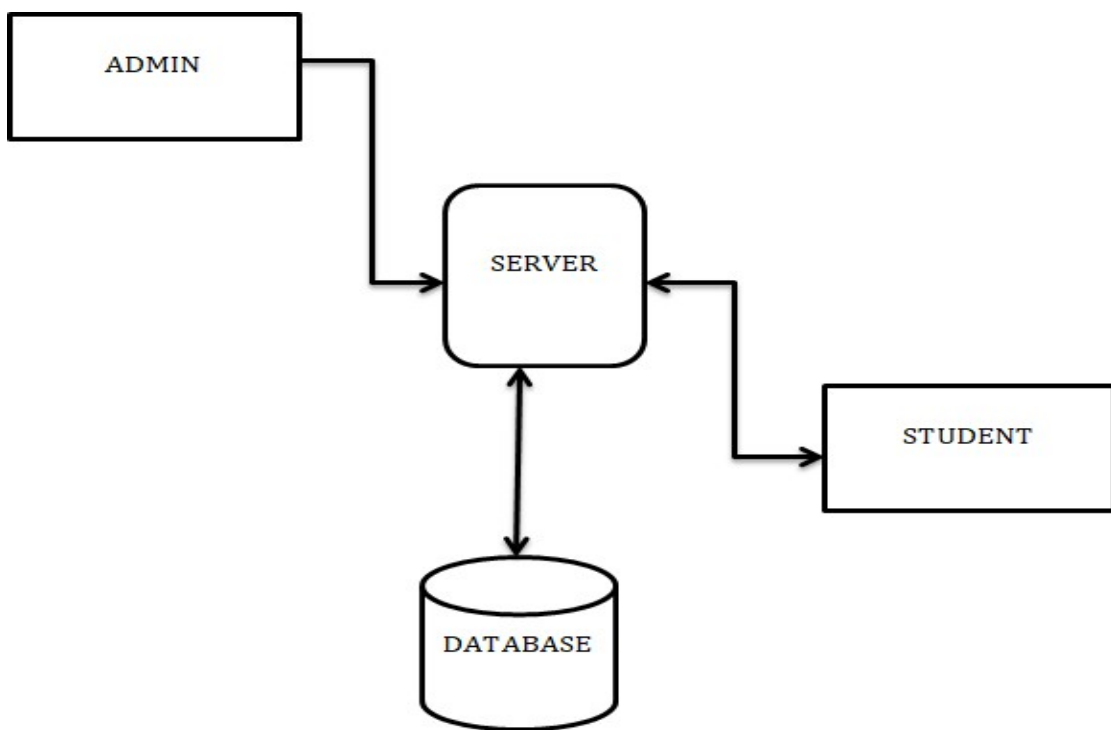
# E-R Diagram:

An Entity Relationship (ER) Diagram is a type of flowchart that illustrate show “entities” such as people, objects or concepts relate to each other within a system.



## DFD:

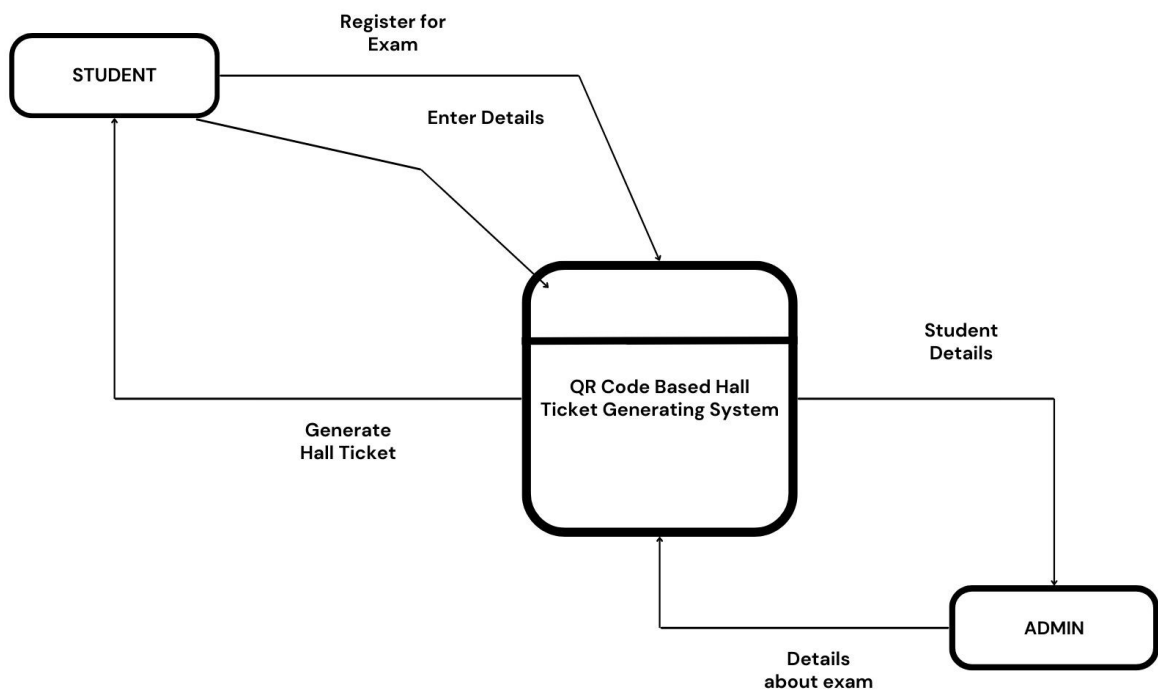
A data flow diagram shows the way information flows through a process or system



Level 0 DFD

# Context Diagram:

Context diagrams show the interactions between a system and other actors (external factors) with which the system is designed to interface.



# Coding:

Home.jsx:-

```
import React from "react";
import Nav from "../Nav";
import Text from "../Text";
import Count from "../Count/Count";
import Card from "../Card/Card";
import pic1 from "../assets/c1.png";
import pic2 from "../assets/c2.png";
import pic3 from "../assets/c3.png";
import pic4 from "../assets/c4.jpg";
import Que from "../Que";
import TeacherCard from "../Card/TeacherCard";
import Footer from "../Footer";
import ProfileCard from "../Profile/ProfileCard";

function Home() {
  return (
    <>
      <Nav />
      <br />
      <br />
      <Text />
      <br />
      <br />
      <br />
      <Count />
      <br />
      <br />
      <h1 className="text-center">Choose Our Top Courses</h1>
      <br />
      <br />
      <div className="flex flex-wrap justify-center gap-6">
        <Card img={pic1} />
        <Card img={pic2} />
        <Card img={pic3} />
      </div>
      <br />
      <br />
      <Que />
      <h1 className="text-center">Our Expert Instructors</h1>
      <br />
      <br />
      <br />
      <div className="flex flex-wrap justify-center gap-6">
        <TeacherCard img={pic4} />
        <TeacherCard img={pic4} />
        <TeacherCard img={pic4} />
      </div>
      <br />
      <br />
      <br />
      <br />
      <br />
      <Footer />
    </>
  );
}

export default Home;
```

## signup.jsx:-

```
import React, { useEffect, useRef } from "react";
import { useNavigate } from "react-router";
import { useAuth } from "../../Auth/AuthProvider";
import { Link } from "react-router-dom";

function Log() {
  const { user, userLog } = useAuth();
  const navigate = useNavigate();
  const userInfo = useRef(null);
  useEffect(() => {
    if (user) {
      navigate("/");
    } else if (!user) {
      navigate("/loginFrom");
    }
  }, [user]);

  const handleLogin = (e) => {
    e.preventDefault();
    console.log("user Log");
    const email = userInfo.current.email.value;
    const password = userInfo.current.password.value;
    const userInfoDetails = {
      email,
      password,
    };
    userLog(userInfoDetails);
  };
}
```

```
return (
  <>
    <div class="bg-gray-100 flex h-screen items-center justify-center p-4">
      <div class="w-full max-w-md">
        <div class="bg-white shadow-md rounded-md p-8">
          

          <h2 class="mt-6 text-center text-3xl font-bold tracking-tight text-gray-900">
            Sign in to your account
          </h2>

          <form class="space-y-6 mt-4" ref={userInfo} onSubmit={handleLogin}>
            <div>
              <label
                for="password"
                class="block text-sm font-medium text-gray-700"
              >
                Password
              </label>
              <div class="mt-1">
                <input
                  name="password"
                  type="password"
                  autocomplete="password"
                  required
                  class="px-2 py-3 mt-1 block w-full rounded-md border border-gray-300 shadow-sm
focus:border-sky-500 focus:outline-none focus:ring-sky-500 sm:text-sm"
                />
              </div>
            </div>
            <div>
              <label
                for="email"
                class="block text-sm font-medium text-gray-700"
              >
                Email
              </label>
              <div class="mt-1">
                <input
                  name="email"
                  type="email"
                  autocomplete="email"
                  required
                  class="px-2 py-3 mt-1 block w-full rounded-md border border-gray-300 shadow-sm
focus:border-sky-500 focus:outline-none focus:ring-sky-500 sm:text-sm"
                />
              </div>
            </div>
            <div>
              <button
                type="submit"
                class="w-full py-3 text-white bg-sky-600 rounded-md shadow-sm"
              >
                Sign in
              </button>
            </div>
          </form>
        </div>
      </div>
    </div>
  </>
)
```





```
<di.v>

    <label
      for="password"
      class="blocktext-smfont-medi.umtext-gray-700"
    >
      Password
    </label>
    <di.vclass="mt-1">
      <i.nput
        i.d="password"
        name="password"
        type="password"
        "autocomplete="password"requi
        .red
        class="px-2py-3mt-1blockw-fullrounded-mdborderborder-gray-300shadow-sm
focus:border-sky-500 focus:outline-none focus:ri.ng-sky-500 sm:text-sm"
      />
    </di.v>
  </di.v>

  <di.v>
    <button
      type="submi.t"
      class="flexw-fulljustify-centerrounded-mdborderborder-transparentbg-sky-400py-2 px-4
text-sm font-mediumtext-whi.teshadow-smhover:bg-opaci.ty-75 focus:outline-none focus:ri.ng-2 focus:ring-sky-
400 focus:ring-offset-2"
    >
      SignIn
    </button>
    <Li.nkto={"/logi.n"}>
      <label
        for="newace"
        class="blocktext-smfont-medi.umtext-blue-700"
      >
        CreateAccount
      </label>
    </Link>
  </di.v>
</form>
</di.v>
</div>
</di.v>
</>

};

exportdefaultLog;
```

### Login.jsx:-

```

import React, { useEffect, useRef } from "react";
import { useNavigate } from "react-router";
import { useAuth } from "../../Auth/AuthProvider";
import { Link } from "react-router-dom";

function Log() {
  const { user, userLog } = useAuth();
  const navigate = useNavigate();
  const userInfo = useRef(null);
  useEffect(() => {
    if (user) {
      navigate("/");
    } else if (!user) {
      navigate("/loginFrom");
    }
  }, [user]);

  const handleLogin = (e) => {
    e.preventDefault();
    console.log("user Log");
    const email = userInfo.current.email.value;
    const password = userInfo.current.password.value;
    const userInfoDetails = {
      email,
      password,
    };
    userLog(userInfoDetails);
  };

  return (
    <div class="bg-gray-100 flex h-screen items-center justify-center p-4">
      <div class="w-full max-w-md">
        <div class="bg-white shadow-md rounded-md p-8">
          

          <h2 class="mt-6 text-center text-3xl font-bold tracking-tight text-gray-900">
            Sign in to your account
          </h2>
        </div>
      </div>
    </div>
  );
}

```

```

<form class="space-y-6 mt-4" ref={userInfo} onSubmit={handleLogin}>
  <div>
    <label
      for="password"
      class="block text-sm font-medium text-gray-700"
    >
      Email
    </label>
    <div class="mt-1">
      <input
        name="email"
        type="email-address"
        autoComplete="email-address"
        required
        class="px-2 py-3 mt-1 block w-full rounded-md border border-gray-300 shadow-sm focus:border-sky-500 focus:outline-none focus:ring-sky-500 sm:text-sm"
      />
    </div>
  </div>
  <div>
    <label
      for="password"
      class="block text-sm font-medium text-gray-700"
    >
      Password
    </label>
    <div class="mt-1">
      <input
        id="password"
        name="password"
        type="password"
        autoComplete="password"
        required
        class="px-2 py-3 mt-1 block w-full rounded-md border border-gray-300 shadow-sm focus:border-sky-500 focus:outline-none focus:ring-sky-500 sm:text-sm"
      />
    </div>
  </div>
  <button
    type="submit"
    class="flex w-full justify-center rounded-md border border-transparent bg-sky-400 py-2 px-4 text-sm font-medium text-white shadow-sm hover:bg-opacity-75 focus:outline-none focus:ring-2 focus:ring-sky-400 focus:ring-offset-2"
  >
    Sign In
  </button>
  <Link to="/login">
    <label
      for="new acc"
      class="block text-sm font-medium text-blue-700"
    >
      Create Account
    </label>
    </Link>
  </div>
</form>
</div>
</div>
)
}

export default Log;
```

### Qr.jsx:-

```
import React, { useEffect, useState } from "react";
import { useAuth } from "../../Auth/AuthProvider";
import { useNavigate } from "react-router";
import { storage } from "../../appwrite/appwrite";
import conf from "../../Conf/Conf";
import { nanoid } from "nanoid";

const Qr = () => {
  const navigate = useNavigate();
  const { user, datanew, getData, userImage, getUserImage } = useAuth();
  const [last, setLast] = useState(null); // State to store the calculated value

  useEffect(() => {
    getData();
    getUserImage();
  }, []);

  useEffect(() => {
    if (!user) {
      navigate('/loginFrom');
    }
  }, [user, navigate]);

  useEffect(() => {
    // Calculate 'last' value when 'datanew' changes
    datanew.map((data, index) => {
      if (user.$id === data.userId) {
        setLast(data.roll % 100); // Update the 'last' state
      }
    });
  }, [datanew]);

  return (
    <div className="bg-white rounded-lg shadow-md p-8 mx-auto my-5 border border-gray-300" style={{
      width: "21cm", minHeight: "29.7cm", padding: "2cm" }}>
      <h2 className="text-2xl font-bold mb-4">Hall Ticket</h2>
      {datanew &&
        datanew.map((data, indx) => {
          if (user.$id === data.userId) {
            return (
              <div key={indx} className="grid grid-cols-2 gap-8">
                <div>Left Side Details </div>
                <div>
                  <h2 className="text-sm font-bold mb-2">Enrollment Number: {data.unique_Id + last}</h2>

```

```


```
<div className="w-1/2">
    <p className="font-semibold">Student Name:</p>
    <p>{data.name}</p>
</div>
<div className="w-1/2">
    <p className="font-semibold">Exam Date:</p>
    <p>{data.email}</p>
</div>
<div>
<div className="flex justify-between mb-4">
<div className="w-1/2">
    <p className="font-semibold">Exam Time:</p>
    <p>{data.stream}</p>
</div>
</div>
<div className="flex justify-between mb-4">
<div className="w-1/2">
    <p className="font-semibold"> Stream </p>
    <p>{data.stream}</p>
</div>
<div className="w-1/2">
    <p className="font-semibold">Roll Number:</p>
    <p>{data.roll}</p>
</div>
</div>
<div className="flex justify-between mb-4">
<div className="w-1/2">
    <p className="font-semibold"> Subject </p>
    <p>{data.sub}</p>
</div>
</div>
<div className="flex justify-between mb-4">
<div className="w-1/2">
    <p className="font-semibold"> Email </p>
    <p>{data.email}</p>
</div>
</div>
<div>
    <p className="font-semibold">Exam Venue:</p>
    <p>{data.pic_id}</p>
</div>
</div>

/* Right Side Pictures */
<div className="relative top-5 right-0 mt-4 mr-4">
    <div className="flex justify-between">
        <img src={`https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=${
            data.unique_Id + last + data.name }`} alt="Image 1" className="w-20 mr-2" />
        <img src={storage.getFilePreview(conf.appwriteBucketId_1, data.pic_id)} alt="Image 2"
            className="w-20 ml-2" />
    </div>
    </div>
    )
}
return null; // Return null if user ID doesn't match
}}}
</div>
};
});
export default Qr;
```


```

## Profile.jsx:-

```
import React, { useEffect, useState } from "react";
import { useAuth } from "../../Auth/AuthProvider";
import { useNavigate } from 'react-router';
import { storage } from "../../appwrite/appwrite";
import conf from "../../Conf/Conf";

function ProfileCard() {
  const navigate = useNavigate();
  const { user, datanew, getData, userImage, getUserImage } = useAuth();
  useEffect(() => {
    getData();
    getUserImage();
  }, []);

  useEffect(() => {
    if (!user) {
      navigate('/loginFrom');
    }
  }, [user, navigate]);


  return (
    <>
    {
      datanew && datanew.map((data, index) => {
        if (user.$id === data.userId) {
          return (
            <div key={index} class="flex items-center h-screen w-full justify-center">

<div class="w-1/5">
          <div class="bg-white shadow-xl rounded-lg py-3">
            <div class="photo-wrapper p-2">
              <img class="w-32 h-32 rounded-full mx-auto" src=
{storage.getFilePreview(conf.appwriteBucketId_1, data.pic_Id)} alt="John Doe"/>
            </div>
            <div class="p-2">
              <h3 class="text-center text-xl text-gray-900 font-medium leading-8">{data.name}</h3>
              <div class="text-center text-gray-400 text-xs font-semibold">
                <p>Student</p>
              </div>
              <table class="text-xs my-3">
                <tbody><tr>
                  <td class="px-2 py-2 text-gray-500 font-semibold">Stream</td>
                  <td class="px-2 py-2">{data.stream}</td>
                </tr>
                <tr>
                  <td class="px-2 py-2 text-gray-500 font-semibold">Roll</td>
                  <td class="px-2 py-2">{data.roll}</td>
                </tr>
                <tr>
                  <td class="px-2 py-2 text-gray-500 font-semibold">Email</td>
                  <td class="px-2 py-2">{data.email}</td>
                </tr>
              </tbody></table>
            </div>
          </div>
          </div>
        )
      }
    )
    </>
  )
}

export default ProfileCard
```

# Testing:

Sign-up & Sign-in:-



## Sign up for an account


**Username**

**Email**

**Password**

**Confirm Password**

[Register Account](#)



## Sign in to your account

**Email**

**Password**

[Sign In](#)

[Create Account](#)

**ABC Institute of Technology** is a premier educational institute in Kolkata. It is a part of XYZ Educational Initiatives. Today, it ranks among the top private engineering college in Kolkata, West Bengal. Most of its eligible programs are NBA accredited. It has received several accolades and rankings from industry leaders. Some of them are NIRF, CAREERS360, India Today, The Week, and Times of India. Narula Institute of Technology stands among the best private engineering colleges in Kolkata, West Bengal through its near-perfect adherence to global quality standards.



|          |                  |              |               |
|----------|------------------|--------------|---------------|
| 500      | 540              | 75           | 1000          |
| STUDENTS | BOOKS & JOURNALS | LABORATORIES | STRONG ALUMNI |

Choose Our Top Courses

## Upload Document:-



### Upload Your Document

Your Full Name

Soumo Dhali

Email

soumodhali1234@gmail.com

Stream

MCA

University Roll

1234567

Subject (Math, Computer)

OPP,C

Your Picture

man-shirt-tie-business...tar-600nw-548848999.jpg

Submit



**Soumo Dhali**

Student

Stream MCA

Roll 1234567

Email soumodhali1234@gmail.com

## Hall Ticket :-

### Hall Ticket

**Enrollment Number:** vqjUVURkhJ67

**Student Name:** Soumo Dhali  
**Exam Date:** 08/07/2024

**Exam Time:**  
14:30

**Stream** MCA  
**Roll Number:** 1234567

**Subject**  
OPP,C

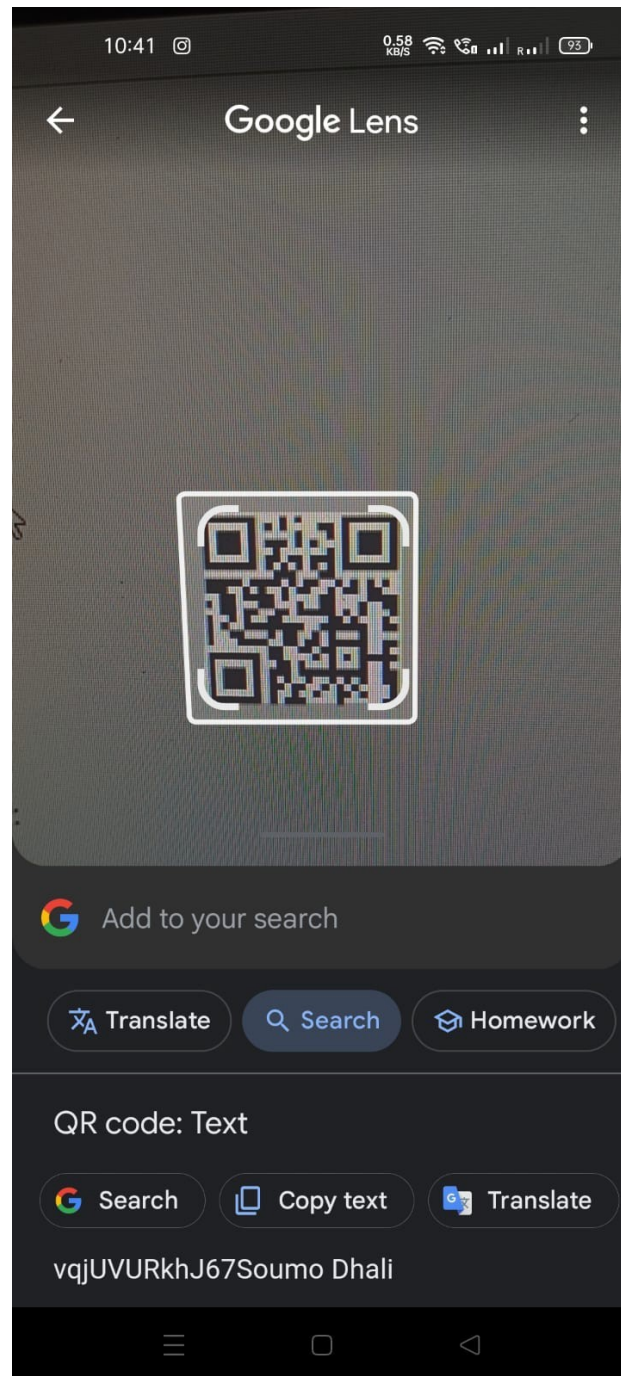
**Email**  
soumodhali1234@gmail.com

**Exam Venue:**  
Kolkata






Scan Result:-



Admin Login:-



Admin Login

Email

admin1234@gmail.com

Password

.....

Sign In

Create Account

Admin Portal:-

| USER DATA   |        |                 |  |            |
|-------------|--------|-----------------|--|------------|
| NAME        | STREAM | ROLL            |  | UNIQUE     |
| SOU MO      | MCA    | 123456777777778 |  | wftbxRWSGE |
| Soumo Dhali | MCA    | 1234567         |  | vqjUVURkhJ |

Software testing is the process of checking the quality, functionality, and performance of a software product before it's released.

***Black box testing***

The tester focuses solely on the external behavior of the software, without having access to its internal source code

***White box testing***

The tester is aware of the internal workings of the product, has access to its source code, and is conducted by making sure that all internal operations are performed according to the specifications

# Bibliography:

- [www.wikipedia.org](http://www.wikipedia.org)
- [www.google.com](http://www.google.com)
- [www.youtube.com](http://www.youtube.com)

## **Conclusion:**

The Hall Ticket Generator using QR Code project represents a significant advancement in the field of examination management. In conclusion, the Hall Ticket Generator using QR Code project showcases the power of technology to improve administrative processes in education, setting a new standard for efficiency and security in examination management.