PROJECT REPORT

ON

”STUDENT TEACHER ATTENDANCE MANAGEMENT SYSTEM”

Submitted to

Sant Gadge Baba Amravati University, Amravati.

In partial fulfillment of the requirements of

M.Sc. (Computer Software) Final Year Examination

Submitted by

Prajwal P. Kadu

Under the guidance of

Mr. Y. V. Hushare

Assistant Professor

(Department of Computer Science)

Shri Shivaji Education Society Amravati’s

SHRI SHIVAJI SCIENCE COLLEGE

Amravati.

2023-2024

CERTIFICATE

This is to certify that the Project Report entitled **STUDENT-TEACHER ATTENDANCE MANAGEMENT SYSTEM** being submitted by **Prajwal P. Kadu** in partial fulfillment for the award of Master of Science in Computer Software (Final Year) **Sant Gadge Baba Amravati University, Amravati** is a record of work carried out for the session 2023-24.

To the best of my knowledge, the matter presented in this project has not been presented earlier for a similar degree/diploma.

Place : Amravati **Project Guide**
Date : **Mr. Y. V. Hushare**

Internal Examiner **External Examiner**

Head
Dept. of Computer Science

DECLARATION

To,
The Principal,
Shri Shivaji Science College,
Amravati.

Respected Sir,

I the undersigned, hereby declare that the Project work entitled **STUDENT-TEACHER ATTENDANCE MANAGEMENT SYSTEM** submitted to **Sant Gadge Baba Amravati University, Amravati** is my independent work. This is my original work and has not been submitted anywhere for any degree/diploma. The system presented herein has not been duplicated from any other source.

I understand that any such copying is liable to be punished in any way the University authority may deem fit.

Thanking You.

Place : Amravati
Date :

Yours Sincerely,

Prajwal P. Kadu
M.Sc. IInd (Sem-IV)

ACKNOWLEDGMENT

We wish to express our sincere thanks to the many persons who helped us to develop our original project work.

First, we express our sincere thanks to Principal **Dr. G. V. Korpe**, Shri Shivaji Science College, Amravati for providing the infrastructure and facilities without which it would have been impossible to complete this hard task.

Foremost this is to**, Dr. M. M. Bhonde**, Head of the Department of Computer Science who has aided us in completing this project report. and I am thankful to **Mr. Y. V. Hushare,** Assistant Professor, Department of Computer Science, Shri Shivaji Science College, Amravati for their constant inspiration and guidance throughout this project work.

Our foremost thanks are to Other Staff, who have guided us in completing this project report, We take the opportunity to express our deep sense of gratitude and wholehearted thanks for his inspiration and guidance throughout this project.

I express my gratitude to all members of the teaching and non-teaching staff of the Department of Computer Science for their cooperation during the course.
Finally, we thank our friends and especially those who helped us in our endeavors.

Place: Amravati
Date:

INDEX

INTRODUCTION

ABSTRACT

The Student-Teacher Attendance Management System (STAMS) is a comprehensive web-based application designed to simplify and streamline the process of managing student attendance in educational institutions. This system automates the recording of student attendance, providing real-time updates to both teachers and students. STAMS aims to enhance efficiency, accuracy, and transparency in attendance management, thereby improving overall academic performance and accountability.

The STAMS project consists of several key modules, including user authentication, class creation, student enrollment, QR code generation, attendance marking, and analytics generation. Users, including teachers and students, can access the system through a user-friendly web interface, making it convenient to use and accessible from any device with internet connectivity.

One of the primary objectives of STAMS is to provide teachers with a convenient platform to create classes, generate QR codes for attendance tracking, and view attendance analytics for individual students. Students, on the other hand, can easily join classes, scan QR codes to mark attendance, and monitor their attendance records.

The software requirements for STAMS include a web server (such as Python Flask), a database management system (such as MongoDB), and front-end technologies (HTML, CSS, JavaScript). Hardware requirements are minimal and include a computer with internet access for hosting the system and devices with a camera for QR code scanning.

STAMS offers several benefits, including:

1. Efficiency: Automation of attendance recording saves time for teachers and students.
2. Accuracy: Real-time attendance tracking reduces errors associated with manual recording.
3. Analytics: Teachers can analyze attendance data to identify trends and patterns, facilitating better decision-making.

INTRODUCTION

In the contemporary educational landscape, the management of student attendance remains a critical aspect of academic administration. Ensuring that students attend classes regularly is not only essential for maintaining discipline but also crucial for monitoring their academic progress and ensuring their success. Traditional methods of attendance management, such as manual recording and paper-based registers, are often labor-intensive, prone to errors, and lack real-time monitoring capabilities.

Recognizing the need for a more efficient and reliable solution, the Student Attendance Management System (STAMS) emerges as a comprehensive web-based application designed to address the challenges associated with traditional attendance tracking methods. By leveraging modern technology, STAMS aims to streamline the process of attendance management, enhance accuracy, and provide valuable insights into student attendance patterns.

Background:

Traditional methods of attendance management rely on manual processes, where teachers manually record attendance in paper registers during each class session. However, this approach is not without its limitations. Manual recording is time-consuming, prone to errors, and lacks real-time visibility into attendance data. Moreover, the reliance on paper-based records makes it challenging to maintain and retrieve attendance information efficiently.

In contrast, the advent of digital technology has paved the way for more sophisticated and automated attendance management systems. These systems offer features such as biometric authentication, RFID-based attendance tracking, and QR code scanning, enabling real-time monitoring and analysis of student attendance data. STAMS builds upon these technological advancements to provide a user-friendly and efficient solution for attendance management in educational institutions.

OBJECTIVES

The objectives of the Student Attendance Management System (STAMS) are as follows:

1. Automation: Implement automated processes for recording and tracking student attendance to reduce manual effort and administrative burden on teachers and staff.

2. Accuracy: Ensure the accuracy and reliability of attendance data by leveraging technology-based solutions that minimize the risk of errors and discrepancies.

3. Transparency: Promote transparency in attendance tracking by providing both teachers and students with access to real-time attendance records and reports.

4. Efficiency: Streamline administrative tasks related to attendance management, such as creating classes, enrolling students, and generating attendance reports, to improve overall operational efficiency.

5. Insights: Generate comprehensive analytics and reports on student attendance patterns and trends to empower educators with actionable insights for enhancing student engagement and academic performance.

6. Integration: Facilitate seamless integration with existing educational systems and platforms to ensure interoperability and maximize usability for teachers, students, and administrators.

7. Security: Implement robust security measures to safeguard sensitive attendance data and ensure compliance with data protection regulations and privacy standards.

8. User Experience: Design an intuitive and user-friendly interface that simplifies the attendance management process for both teachers and students, enhancing overall user experience and satisfaction.

9. Scalability: Build a scalable and adaptable system architecture capable of accommodating future growth and expansion of educational institutions without compromising performance or functionality.

SCOPE OF STUDY

The scope of STAMS encompasses the following key functionalities:

1. Functional Scope:
   - Design and implementation of user authentication and authorization mechanisms for teachers and students.
   - Creation and management of classes by teachers, including adding students and generating class-specific QR codes.
   - Joining classes by students using class codes provided by teachers.
   - Recording and marking attendance for students by teachers using QR code scanning.
   - Generation of attendance reports and analytics for teachers to track student attendance patterns and performance.
   - Profile management functionality for users to update their personal information.

2. Analytical Scope:
   - Analysis of student attendance data to identify trends, patterns, and outliers using statistical methods and data visualization techniques.
   - Generation of attendance reports at class and individual student levels to provide insights into attendance behavior.
   - Evaluation of the effectiveness of attendance management strategies and interventions in improving student engagement and academic performance.

3. Usability Scope:
   - Assessment of the usability and user experience of the STAMS application through user testing and feedback collection.
   - Iterative refinement of the user interface and features based on user input to enhance usability and satisfaction.
   - Investigation of accessibility considerations to ensure inclusivity and usability for users with diverse needs and abilities.

SIGNIFICANCE OF STUDY

STAMS holds significant importance for educational institutions, teachers, and students alike. By replacing manual attendance tracking methods with an automated and digital solution, STAMS offers the following benefits:

- Improved Efficiency: Automation of attendance recording saves time and effort for teachers, allowing them to focus on instructional activities.

- Enhanced Accuracy: Real-time attendance tracking reduces the risk of errors associated with manual recording, ensuring the reliability of attendance data.

- Transparency and Accountability: Providing students with access to their attendance records promotes transparency and encourages accountability for their learning.

- Data-Driven Decision-Making: Attendance analytics and reports enable teachers to identify patterns and trends in student attendance, facilitating informed decision-making and intervention strategies.

- Seamless Integration: STAMS can be seamlessly integrated into existing educational workflows, complementing other learning management systems and administrative tools.

REQUIREMENT AND ANALYSIS

PURPOSE

The purpose of the Student Attendance Management System (STAMS) project is to develop a comprehensive and efficient solution for managing and monitoring student attendance in educational institutions. The primary objectives of STAMS are:

1. Automation: To automate the process of recording student attendance using modern technology, such as QR code scanning, to reduce the manual effort required by teachers and administrators.

2. Accuracy: To ensure accurate and reliable tracking of student attendance, minimizing errors and discrepancies in attendance records.

3. Transparency: To provide transparency in attendance management by allowing teachers, students, and parents to access attendance information easily through an online platform.

4. Analytics: To generate attendance reports and analytics that offer insights into attendance patterns, trends, and performance metrics, enabling data-driven decision-making by teachers and school administrators.

5. Engagement: To promote student engagement and accountability by monitoring attendance regularly and intervening early to address attendance issues.

6. Compliance: To facilitate compliance with regulatory requirements and auditing standards by maintaining well-documented and organized attendance records.

7. Parental Involvement: To encourage parental involvement in monitoring and supporting their child's attendance and academic progress, fostering a collaborative relationship between parents, teachers, and students.

PROJECT SCOPE

The scope of the Student Attendance Management System (STAMS) project encompasses the following key aspects:

1. Functional Scope:

- Registration: Allow students, teachers, and administrators to register their accounts securely.
- Login: Provide authenticated access to the system based on user roles (student, teacher).
- Attendance Tracking: Enable teachers to mark attendance for students using QR code scanning or manual entry.
- Classroom Management: Allow teachers to create and manage virtual classrooms, including adding students, scheduling classes, and generating attendance reports.
- Analytics and Reporting: Generate attendance reports and analytics to provide insights into student attendance patterns, trends, and performance metrics.
- Profile Management: Enable users to update their profile information, including personal details, contact information, and preferences.
- Notification System: Implement a notification system to alert users about important updates, reminders, and announcements related to attendance.

2. Non-Functional Scope:

- Security: Implement robust authentication and authorization mechanisms to protect user data and ensure privacy and confidentiality.
- Usability: Design an intuitive and user-friendly interface that facilitates easy navigation and interaction for users of all technical backgrounds.
- Scalability: Develop the system architecture in a scalable manner to accommodate growth in the number of users and data volume over time.
- Performance: Optimize system performance to ensure fast response times and minimal downtime during peak usage periods.
- Compatibility: Ensure compatibility with a wide range of devices, browsers, and operating systems to support accessibility for users across different platforms.
- Reliability: Build a reliable and stable system that operates smoothly under normal conditions and can recover quickly from any failures or disruptions.

EXISTING SYSTEM

The existing system refers to the methods or tools currently in place for managing student attendance before the implementation of the Student Attendance Management System (STAMS). In many educational institutions, traditional methods are used for recording attendance, which often involves manual processes and paper-based systems. Some common aspects of the existing system include:

1. Manual Attendance Registers: Teachers typically maintain physical attendance registers where they mark the presence or absence of students during each class session.

2. Roll Call: Teachers may perform roll calls at the beginning of each class to verbally call out the names of students and note their attendance status.

3. Proxy Attendance: In some cases, students may attempt to manipulate attendance records by having their peers mark them as present even when they are absent.

4. Data Entry and Management: Attendance data collected manually needs to be entered into digital systems for record-keeping and reporting purposes. This process can be time-consuming and error-prone.

5. Limited Reporting and Analysis: The existing system may lack robust reporting and analytics capabilities, making it difficult for administrators and teachers to gain insights into attendance trends, patterns, and student performance.

6. Communication: Communication between teachers, students, and administrators regarding attendance-related matters may rely on conventional methods such as email, phone calls, or in-person meetings.

7. Access and Security: Access to attendance records and sensitive student information may not be adequately controlled, posing security and privacy risks.

8. Scalability and Efficiency: Manual attendance management processes may become increasingly inefficient and challenging to scale as the number of students and classes grows.

PROPOSED SYSTEM

The proposed system, the Student Attendance Management System (STAMS), aims to address the limitations and challenges of the existing manual attendance management processes by introducing a comprehensive, automated, and user-friendly solution. The proposed system leverages modern technologies to streamline attendance tracking, improve data accuracy, enhance reporting capabilities, and facilitate communication between stakeholders. Key features and components of the proposed system include:

1. Automated Attendance Tracking: STAMS automates the process of recording student attendance using various methods such as QR code scanning, biometric authentication, or online check-ins. This reduces the reliance on manual methods and minimizes the chances of errors or discrepancies in attendance records.

2. Centralized Database: The system maintains a centralized database to store attendance records, student information, class schedules, and other relevant data. This ensures data consistency, integrity, and accessibility for authorized users.

3. Real-time Monitoring and Alerts: Teachers and administrators can monitor attendance in real time and receive alerts for late arrivals, absences, or other attendance-related issues. This proactive approach allows for timely intervention and follow-up actions as needed.

4. Analytics and Reporting: STAMS provides robust analytics and reporting tools to analyze attendance patterns, trends, and student performance metrics. Teachers and administrators can generate customizable reports, charts, and graphs to gain insights into attendance behavior and make data-driven decisions.

5. Mobile Accessibility: The system is accessible via web and mobile applications, allowing users to view attendance records, submit attendance updates, and communicate with stakeholders from any location and device.

6. Integration with Student Information Systems (SIS): STAMS seamlessly integrates with existing student information systems used by educational institutions, ensuring compatibility and interoperability with other administrative systems and processes.

7. Secure Authentication and Access Control: The system implements secure authentication mechanisms such as two-factor authentication (2FA) or single sign-on (SSO) to verify user identities and restrict unauthorized access to sensitive data.

SYSTEM OVERVIEW

The Student Attendance Management System (STAMS) is an innovative software solution designed to streamline and automate the process of tracking student attendance in educational institutions. STAMS leverages modern technologies to provide a comprehensive platform for managing attendance records, facilitating communication between stakeholders, and generating insightful analytics. The system comprises several key components and features:

1. User Management: STAMS allows administrators to manage user accounts for teachers, students, and other staff members. Each user has specific roles and permissions, dictating their access to various features and functionalities within the system.

2. Classroom Management: The system enables teachers to create and manage virtual classrooms, including defining class schedules, assigning subjects, and generating unique class codes for student enrollment.

3. Attendance Tracking: STAMS offers multiple methods for recording student attendance, such as QR code scanning, biometric authentication, or manual check-ins. This automated process reduces the administrative burden on teachers and improves the accuracy of attendance records.

4. Real-time Monitoring: Teachers and administrators can monitor attendance in real-time, receiving notifications for late arrivals, absences, or other attendance-related issues. This proactive approach allows for timely intervention and follow-up actions as needed.

5. Analytics and Reporting: STAMS provides powerful analytics and reporting tools to analyze attendance data, identify trends, and generate customized reports. Teachers and administrators can gain valuable insights into student attendance behavior, enabling data-driven decision-making and intervention strategies.

6. Mobile Accessibility: The system is accessible via web and mobile applications, allowing users to view attendance records, submit updates, and communicate with stakeholders from any location and device.

7. Integration with Student Information Systems (SIS): STAMS seamlessly integrates with existing student information systems used by educational institutions, ensuring compatibility and interoperability with other administrative systems and processes.

8. Security and Data Privacy: STAMS prioritizes the security and privacy of user data, implementing robust encryption, secure authentication mechanisms, and access controls to protect sensitive information from unauthorized access or data breaches.

IMPLEMENTATION ISSUE

IMPLEMENTATION ISSUE

During the implementation of the Student Attendance Management System (STAMS), several key issues may arise that need to be addressed effectively to ensure the successful deployment and operation of the system. Some of the potential implementation issues include:

1. Integration with Existing Systems: One of the primary challenges is integrating STAMS with the existing infrastructure and systems used by educational institutions, such as student information systems (SIS), learning management systems (LMS), or HR systems. Ensuring seamless data exchange and interoperability between STAMS and other systems requires careful planning, coordination, and possibly custom development.

2. User Adoption and Training: Introducing a new attendance management system may face resistance from users who are accustomed to traditional methods or unfamiliar with technology. Providing comprehensive training and support to teachers, students, and administrators is essential to promote user adoption and ensure that everyone understands how to use the system effectively.

3. Technical Infrastructure: STAMS relies on robust technical infrastructure, including servers, databases, networking, and security measures, to operate efficiently and securely. Setting up and maintaining this infrastructure may pose challenges in terms of cost, scalability, reliability, and compliance with regulatory requirements.

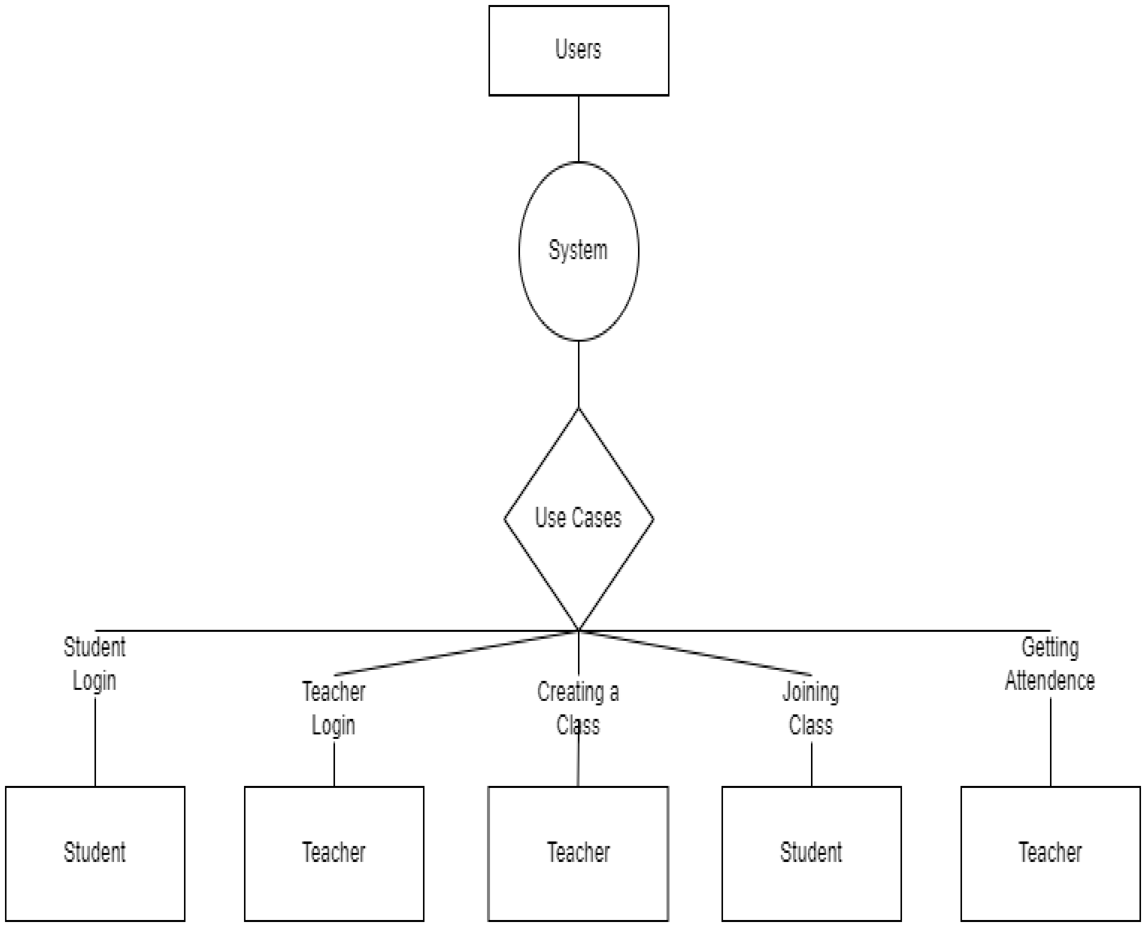
4. Customization and Configuration: Educational institutions have diverse requirements and preferences regarding attendance policies, reporting formats, and system configurations. Customizing STAMS to accommodate these variations while maintaining consistency and usability across different user groups can be complex and time-consuming.

5. Data Security and Privacy: Managing sensitive student data, such as attendance records, requires stringent security and privacy measures to protect against unauthorized access, data breaches, or compliance violations. Implementing robust encryption, access controls, and data governance policies is crucial to safeguarding the confidentiality and integrity of student information.

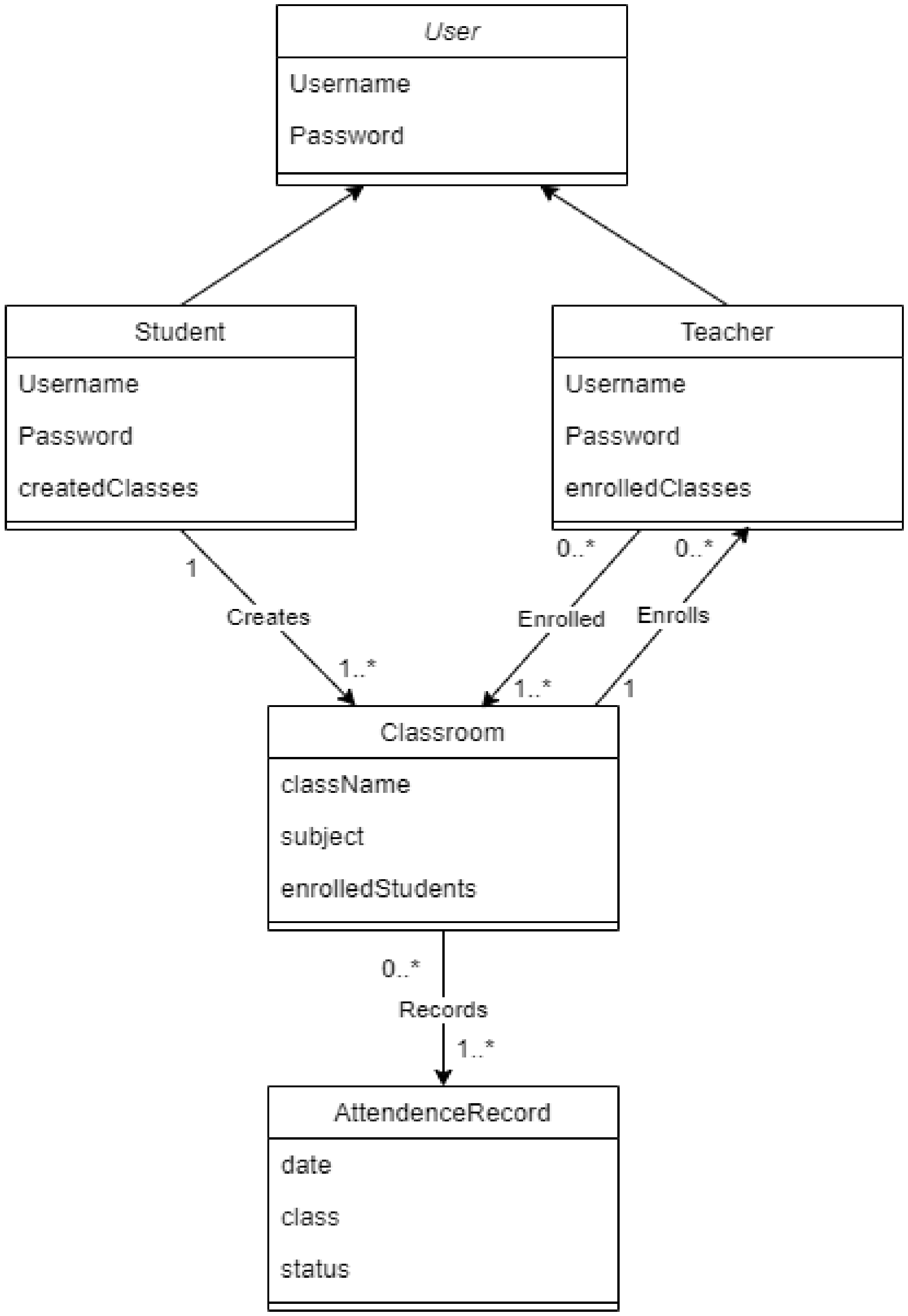
6. User Feedback and Iterative Improvements: Continuous feedback from users, administrators, and other stakeholders is essential for identifying usability issues, functionality gaps, or performance bottlenecks in STAMS. Establishing mechanisms for collecting feedback, prioritizing enhancement requests, and implementing iterative improvements is vital for ensuring the long-term success and satisfaction of system users.

SYSTEM DESIGN

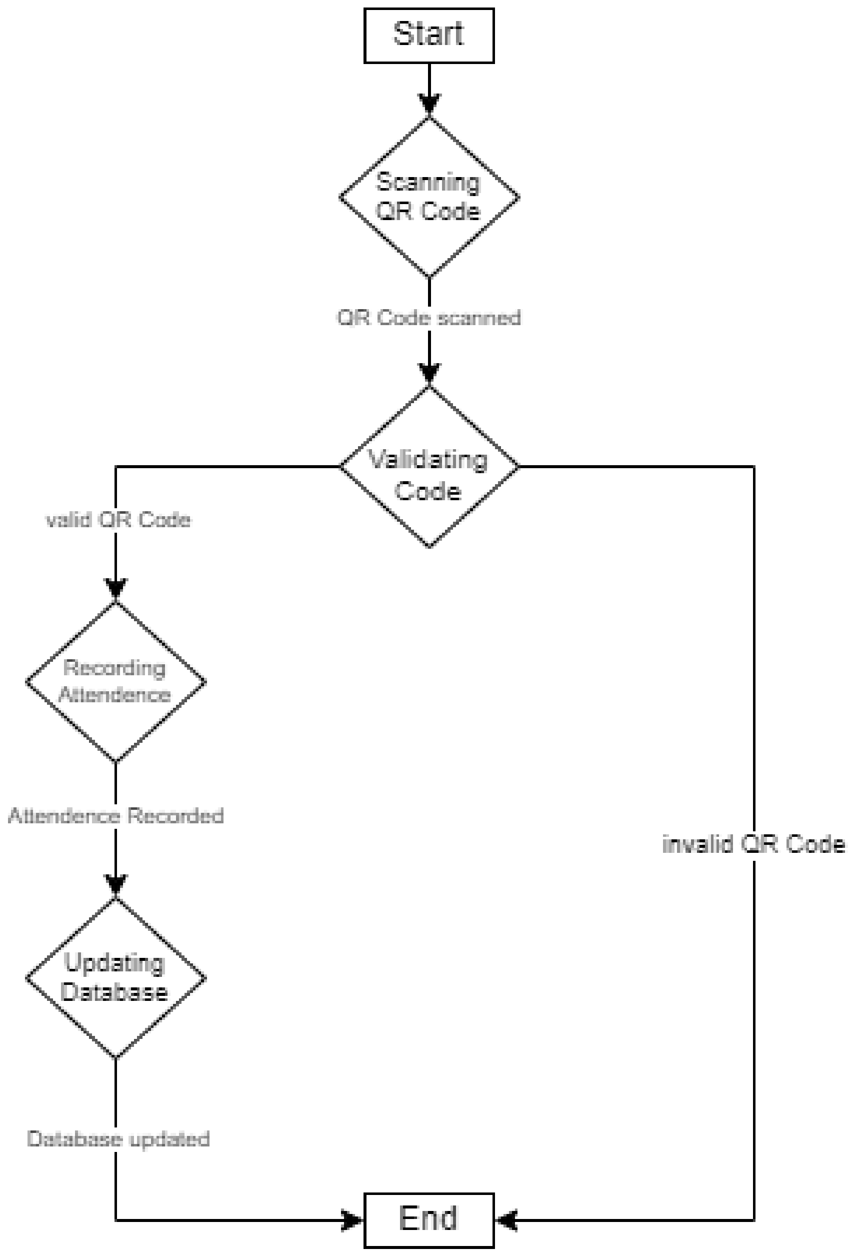
USE CASE DIAGRAM



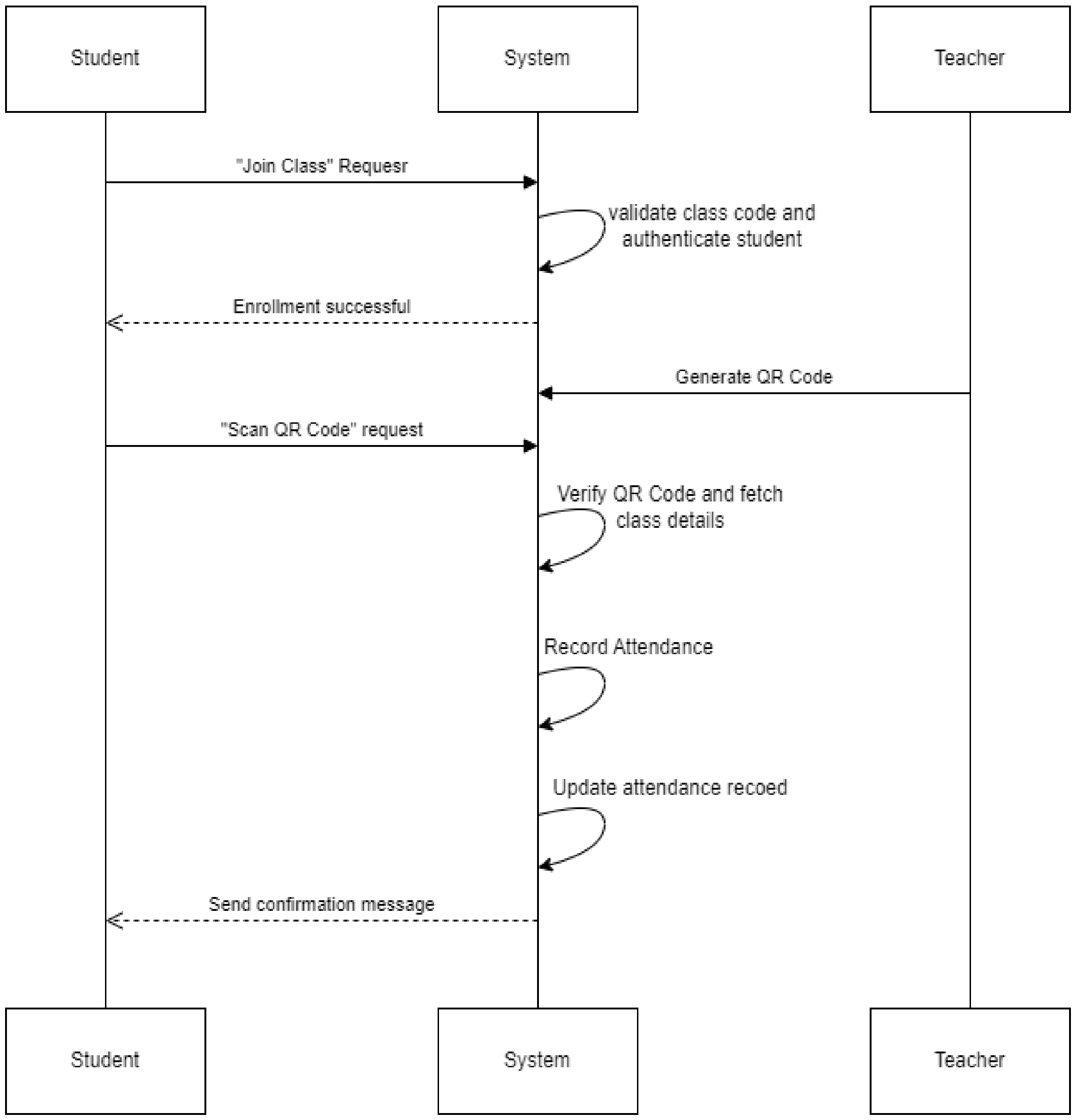
CLASS DIAGRAM



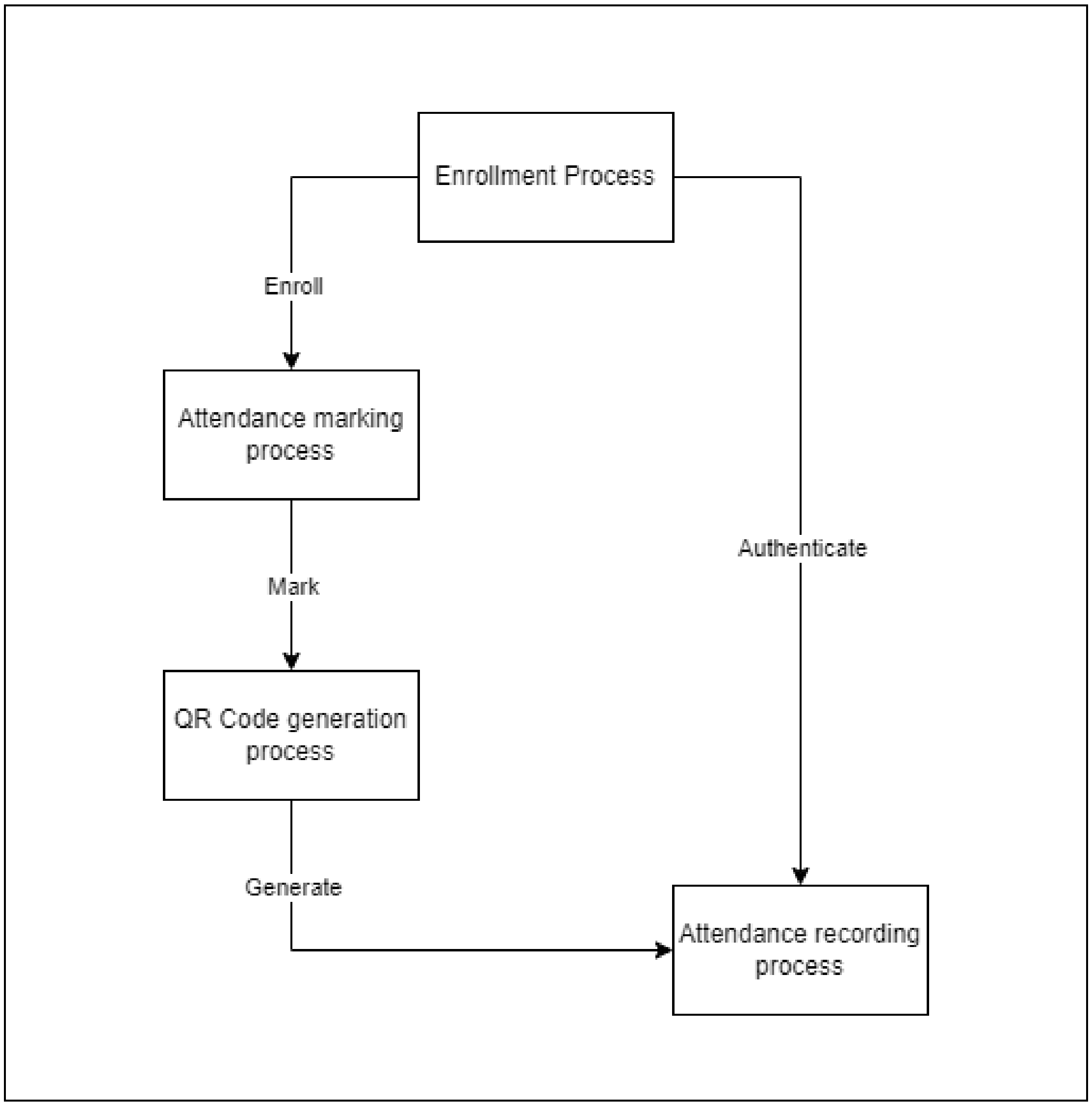
ACTIVITY DIAGRAM



SEQUENCE DIAGRAM

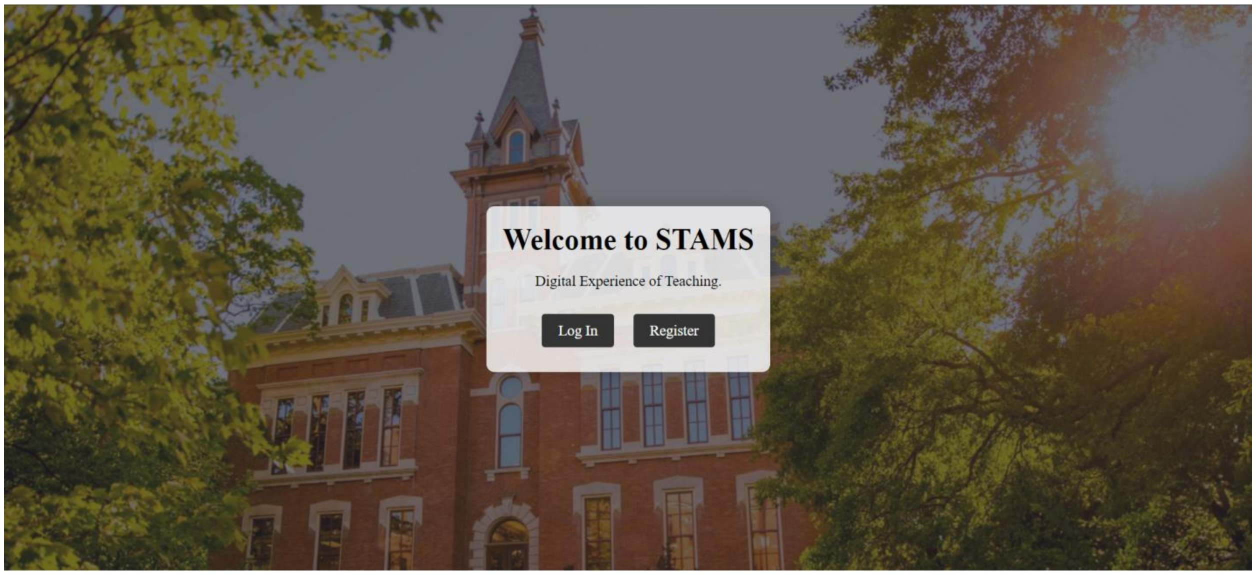


DATA FLOW DIAGRAM

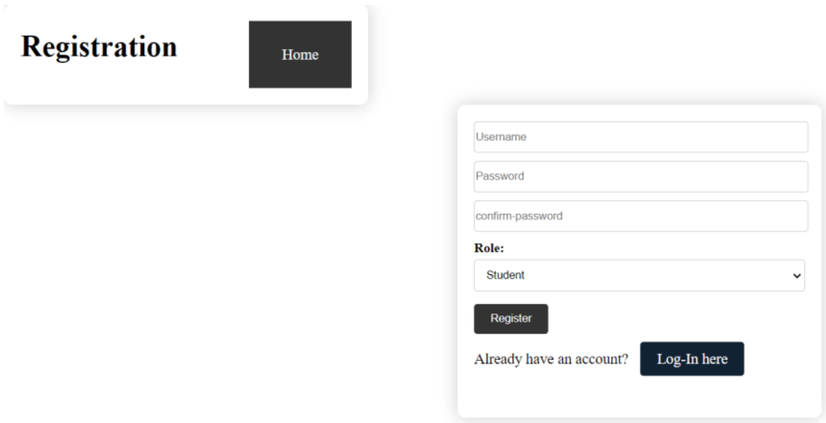

Enrollment Process
Enroll
Attendance marking process
Mark
QR Code generation process
Generate
Authenticate
Attendance recording process

USER SCREENS

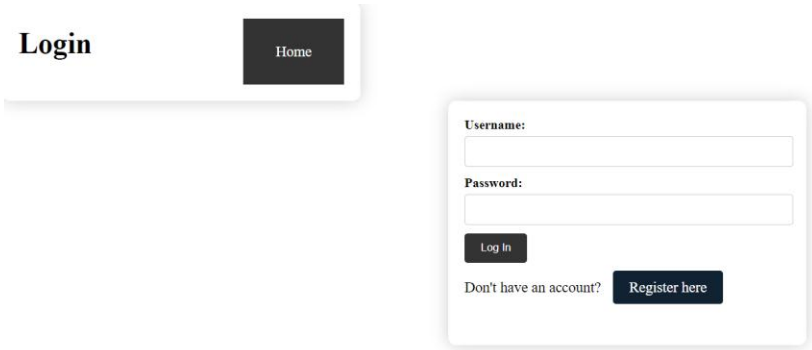
- **Home Page**



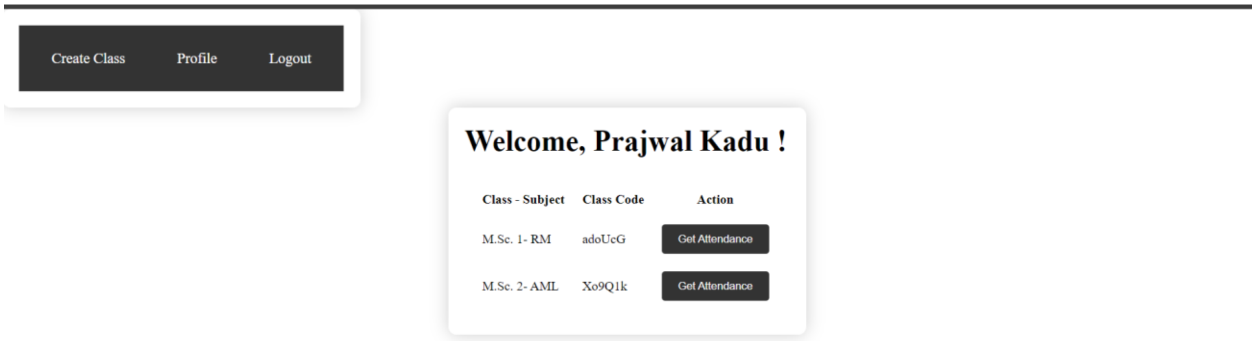
- **Registration Page**



- **Login Page**



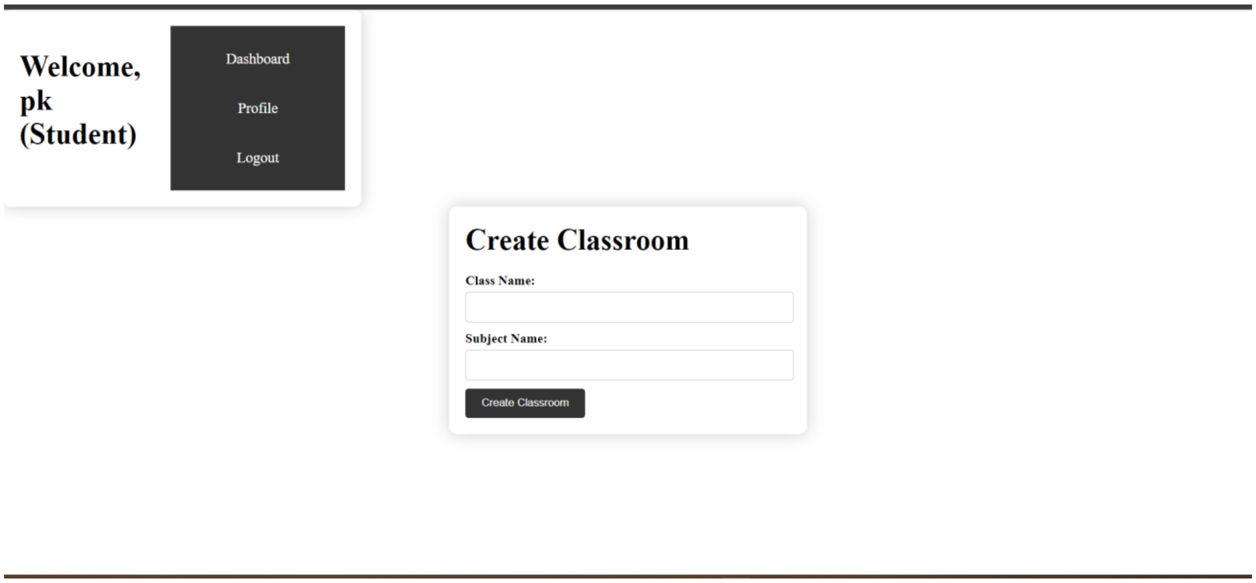
- **Teacher Dashboard Page**



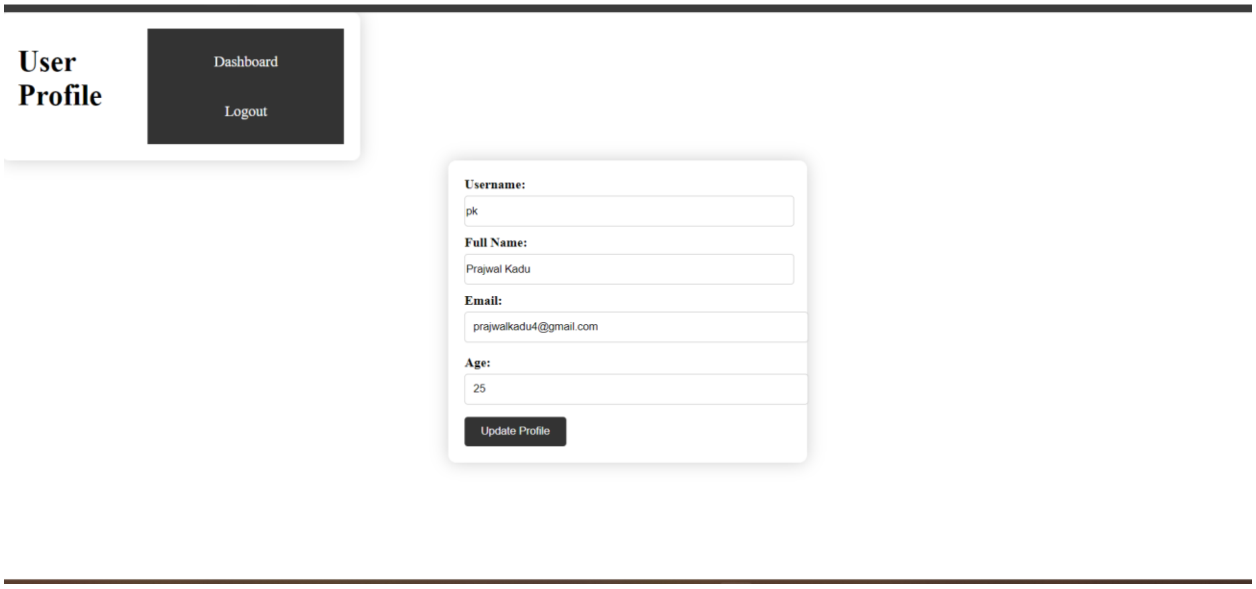
Welcome, Prajwal Kadu !

Class - Subject	Class Code	Action
M.Sc. 1- RM	adoUcG	Get Attendance
M.Sc. 2- AML	Xo9Q1k	Get Attendance

- **Create Classroom Page**



- **Profile Update Page**



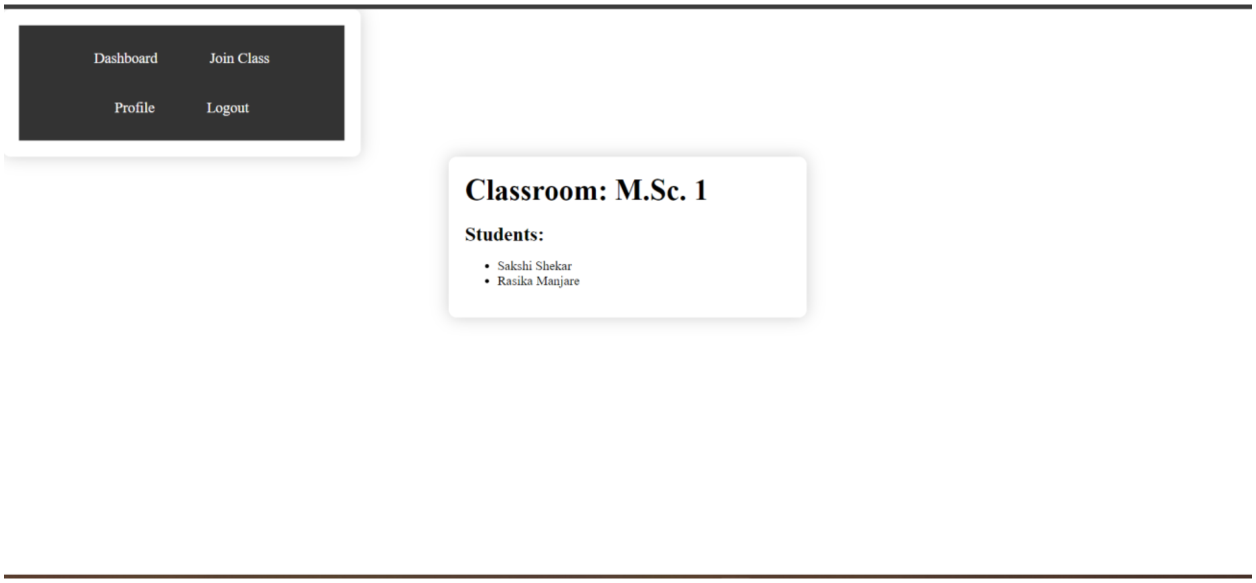
Username: pk

Full Name: Prajwal Kadu

Email: prajwalkadu4@gmail.com

Age: 25

- **Classroom Enrollment Page**



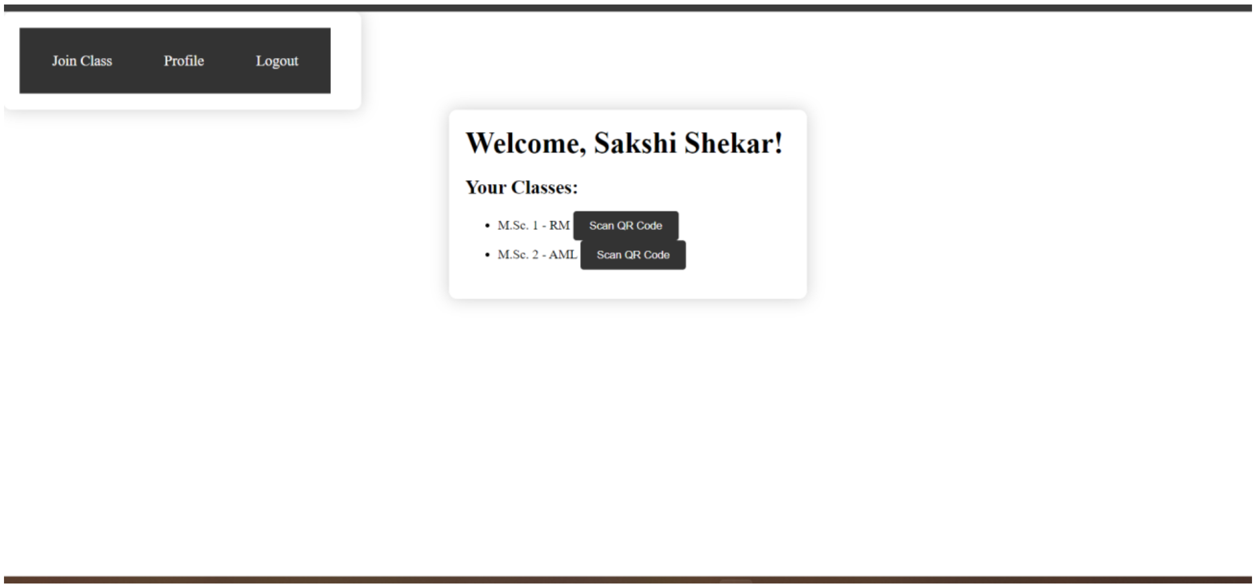
Classroom: M.Sc. 1

Students:

- Sakshi Shekar
- Rasika Manjare

- **Attendance QR Code Page**

- **Student Dashboard Page**

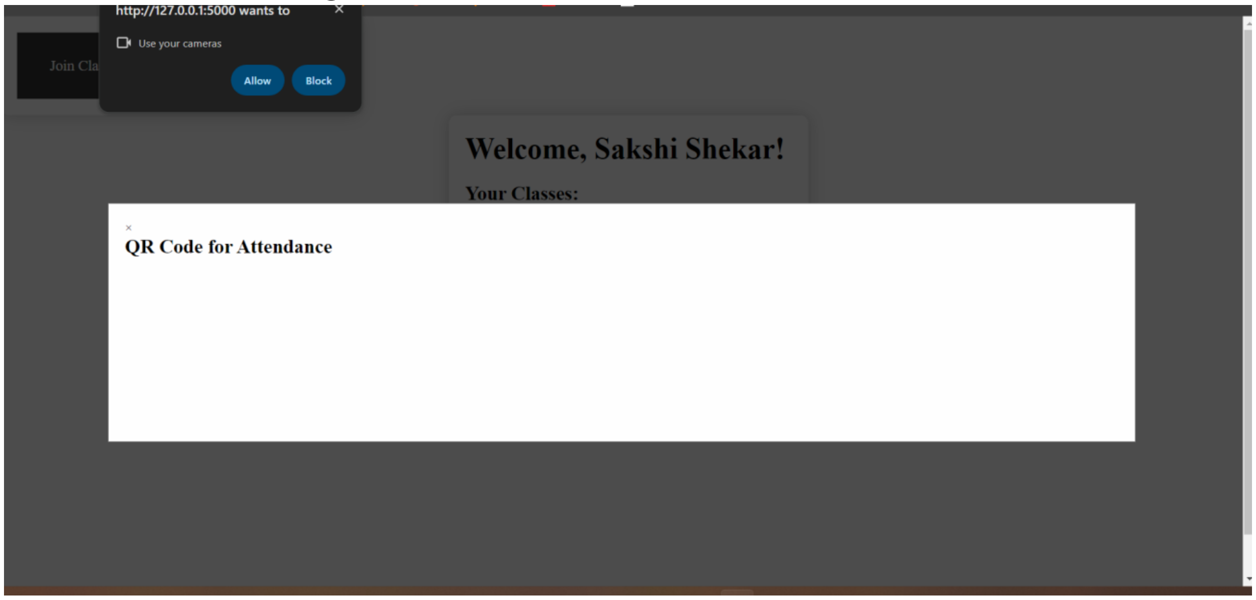


Welcome, Sakshi Shekar!

Your Classes:

- M.Sc. 1 - RM
- M.Sc. 2 - AML

- **Scan QR Code Page**



Welcome, Sakshi Shekar!

Your Classes:

QR Code for Attendance

CODING

- **Main.py**

```
from flask import Flask, render_template, request, redirect, url_for, flash, session, send_file, jsonify
from pymongo import MongoClient
from werkzeug.security import generate_password_hash, check_password_hash
import random
import string
import qrcode
from io import BytesIO
from datetime import datetime, timedelta

app = Flask(__name__)
app.secret_key = 'your_secret_key'

# MongoDB configuration
client = MongoClient("mongodb://localhost:27017/")
db = client["STAMS"]
users = db["users"]
classrooms = db["classrooms"]
qr_codes = db["qr-codes"]


@app.route("/")
def index():
    return render_template("index.html")

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        confirm_password = request.form['confirm-password']
        role = request.form['role']

        if password != confirm_password:
            flash("Passwords do not match.", "error")
            return redirect(url_for('register'))

        hashed_password = generate_password_hash(password)

        if users.find_one({'username': username}):
            flash("Username already exists.", "error")
            return redirect(url_for('register'))
        users.insert_one({'username': username, 'password': hashed_password, 'role': role})
        flash("Registration successful. You can now log in.", "success")
        return redirect(url_for('login'))

    return render_template('registration.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        user = users.find_one({'username': username})
        if user and check_password_hash(user['password'], password):
            session['username'] = username
            if user['role'] == 'student':
                return redirect(url_for('student_dashboard'))
            elif user['role'] == 'teacher':
                return redirect(url_for('teacher_dashboard'))
        else:
            flash("Invalid username or password.", "error")

    return render_template('login.html')

@app.route("/teacher/dashboard")
def teacher_dashboard():
    if "username" not in session:
        return redirect(url_for("login"))

    username = session.get("username")
    user_data = users.find_one({"username": username})
    teacher_username = user_data.get("full_name", "")
    created_classes = classrooms.find({"teacher_username": username})

    return render_template("teacher_dashboard.html", user={"username": teacher_username},
created_classes=created_classes)

@app.route("/student/dashboard")
def student_dashboard():
    if "username" not in session:
        return redirect(url_for("login"))

    username = session.get("username")
    user_data = users.find_one({"username": username})
    student_name = user_data.get("full_name", "")

    joined_classes = classrooms.find({"students": username})

    return render_template("student_dashboard.html", user={"username": student_name},
joined_classes=joined_classes)

@app.route("/dashboard")
def dashboard():
    if 'username' not in session:
        flash("Please log in to access this page.", "error")
        return redirect(url_for('login'))
    user = users.find_one({'username': session['username']})
    user_role = user.get("role", "student")
    if user_role == "teacher":
        return redirect(url_for("teacher_dashboard"))
    elif user_role == "student":
        return redirect(url_for("student_dashboard"))

@app.route("/profile", methods=["GET", "POST"])
def profile():
    if 'username' not in session:
        flash("Please log in to access this page.", "error")
        return redirect(url_for('login'))
    user = users.find_one({'username': session['username']})

    if request.method == "POST":
        full_name = request.form["full_name"]
        email = request.form["email"]
        age = int(request.form["age"])

        users.update_one(
            {'username': session['username']},
            {"$set": {
                "full_name": full_name,
                "email": email,
                "age": age
            }}
        )
        flash("Profile updated successfully", "success")

    return render_template("profile.html", user=user)

@app.route("/logout")
```

```
def logout():
    session.pop('username', None)
    flash("Logged out successfully", "success")
    return redirect(url_for("index"))

def generate_unique_code():
    code_length = 6  # You can adjust the length of the code as needed
    characters = string.ascii_letters + string.digits
    code = ''.join(random.choice(characters) for _ in range(code_length))
    return code

@app.route("/create_class", methods=["GET", "POST"])
def create_classroom():
    user = users.find_one({'username': session['username']})
    if request.method == "POST":
        class_name = request.form["class_name"]
        subject_name = request.form["subject_name"]
        class_code = generate_unique_code()  # Implement your own code generation logic
        teacher_username = session['username']

        classrooms.insert_one({
            "class_name": class_name,
            "subject_name": subject_name,
            "class_code": class_code,
            "teacher_username": teacher_username,
            "students": []  # Initially, the classroom has no students
        })
        flash("Classroom created successfully", "success")
        return redirect(url_for("teacher_dashboard"))

    return render_template("create_classroom.html", user=user)

@app.route("/join_class", methods=["GET", "POST"])
def join_classroom():
    user = users.find_one({'username': session['username']})
    if request.method == "POST":
        class_code = request.form["class_code"]
        student_username = session['username']

        classroom = classrooms.find_one({"class_code": class_code})
        if classroom:
            # Check if the student is not already in the classroom
            if student_username not in classroom["students"]:
                classrooms.update_one(
                {"class_code": class_code},
                {"$push": {"students": student_username}}
            )
            flash("Joined classroom successfully", "success")
        else:
            flash("You are already part of this classroom", "info")
    else:
        flash("Invalid class code", "error")

    return render_template("join_classroom.html", user=user)

@app.route("/classroom/<class_code>")
def classroom(class_code):
    classroom_data = classrooms.find_one({"class_code": class_code})

    # Fetch the usernames as an array from the classroom document
    usernames = classroom_data.get("students", [])

    # Use the usernames to fetch the full names from the "users" collection
    students = []

    for username in usernames:
        user_data = users.find_one({"username": username})
        if user_data:
            students.append(user_data.get("full_name", ""))

    return render_template("classroom.html", classroom=classroom_data, students=students)

@app.route("/generate_qr_code/<class_code>")
def generate_qr_code(class_code):
    classroom_data = classrooms.find_one({"class_code": class_code})
    if classroom_data:
        class_name = classroom_data["class_name"]
        subject_name = classroom_data["subject_name"]
        teacher_name = classroom_data["teacher_username"]
        date = datetime.now().date()
        secret_code = generate_unique_code()  # Implement a function to generate a secret code

        qr_data = f"{class_name}&{subject_name}&{teacher_name}&{date}&{secret_code}"

        qr = qrcode.QRCode(
            version=1,
            error_correction=qrcode.constants.ERROR_CORRECT_L,
            box_size=10,
            border=4,
        )
        qr.add_data(qr_data)
        qr.make(fit=True)

        img = qr.make_image(fill_color="black", back_color="white")

        img_byte_array = BytesIO()
        img.save(img_byte_array, format="PNG")  # Use format argument here

        qr_code_details = {
            "class_code": class_code,
            "qr_data": qr_data,
            "secret_code": secret_code,
            "expiration_time": datetime.now() + timedelta(seconds=15),  # Expiration time set to 15 seconds
        }
        qr_codes.insert_one(qr_code_details)

        return send_file(
            BytesIO(img_byte_array.getvalue()),
            mimetype="image/png",
        )
    else:
        flash("Class not found.", "error")
        return redirect(url_for("teacher_dashboard"))

@app.route("/mark_attendance", methods=['POST'])
def mark_attendance():
    data = request.json
    class_name = data['className']
    subject_name = data['subjectName']
    username = data['studentUsername']
    qr_code_data = data['qrCodeData']

    # Check if the class exists in the database
    class_doc = classrooms.find_one({"class_name": class_name})
    if not class_doc:
        return jsonify({"message": "Class not found"}), 404

    # Extract necessary information from the QR code data
    class_name_qr, subject_name_qr, teacher_name, date, secret_code = qr_code_data.split('&')

    # Check if the date in the QR code matches today's date
    current_date = datetime.now().date()
    if current_date.strftime('%Y-%m-%d') != date:
        return jsonify({"message": "QR code date does not match today's date"}), 400


    # Check if the student has already been marked for today
    class_collection = db[class_name]
    existing_record = class_collection.find_one({"username": username, "date": date, "subject_name": 
subject_name})
    if existing_record:
        return jsonify({"message": "Attendance already recorded for today"}), 400

    # Record the attendance
    class_collection.insert_one({"username": username, "date": date, "subject_name": subject_name})
    return jsonify({"message": "Attendance recorded successfully"}), 200


if __name__ == "__main__":
    app.run(debug=True)
```

- **Index.html**

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>STAMS</title>
   <link rel="stylesheet" type="text/css" href="/static/styles.css">
</head>
<body>

<div class="background-image">
   <div class="container">
      <h1>Welcome to STAMS</h1>
      <p>Digital Experience of Teaching.</p>
      <a href="/login" class="button">Log In</a>
      <a href="/register" class="button">Register</a>
   </div>
</div>

</body>
</html>
```

- **Registration.html**

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Registration</title>
    <link rel="stylesheet" type="text/css" href="/static/styles.css">
</head>
<body>

    <div class="header">
        <h1>Registration</h1>
        <div class="navbar">
            <a href="/" class="button">Home</a>
        </div>
    </div>

    <div class="container">
        <form method="POST">
            <input type="text" name="username" placeholder="Username" required>
            <input type="password" name="password" placeholder="Password" required>
            <input type="password" name="confirm-password" placeholder="confirm-password" required>
            <label for="role">Role:</label>
            <select name="role" id="role">
                <option value="student">Student</option>
                <option value="teacher">Teacher</option>
            </select>
            <button type="submit">Register</button>
        </form>
        <p>Already have an account? <a href="/login" class="button2">Log-In here</a></p>
    </div>
</body>
</html>
```

- **Login.html**

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" type="text/css" href="/static/styles.css"> <!-- Add your CSS file if needed -->
</head>
<body>

    <div class="header">
        <h1>Login</h1>
        <div class="navbar">
            <a href="/" class="button">Home</a>
        </div>
    </div>

    <div class="container">
        <form method="POST">
            <label for="username">Username:</label>
            <input type="text" name="username" id="username" required>

            <label for="password">Password:</label>
            <input type="password" name="password" id="password" required>

            <button type="submit">Log In</button>
        </form>
        <p>Don't have an account? <a href="/register" class="button2">Register here</a></p>
    </div>
</body>
</html>
```

- **Profile.html**

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>User Profile</title>
   <link rel="stylesheet" type="text/css" href="/static/styles.css">
</head>
<body>

   <div class="header">
      <h1>User Profile</h1>
      <div class="navbar">
         <a href="/dashboard" class="button">Dashboard</a>
         <a href="/logout" class="button">Logout</a>
      </div>
   </div>

<div class="container">
   <form method="POST">
      <label for="username">Username:</label>
      <input type="text" name="username" value="{{ user.username }}" readonly>

      <label for="full_name">Full Name:</label>
      <input type="text" name="full_name" value="{{ user.full_name }}" required>

      <label for="email">Email:</label>
      <input type="email" name="email" value="{{ user.email }}" required>

      <label for="age">Age:</label>
      <input type="number" name="age" value="{{ user.age }}" required>

      <button type="submit">Update Profile</button>
   </form>
</div>

</body>
</html>
```

- **Student_dashboard.html**

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Student Dashboard</title>
   <link rel="stylesheet" type="text/css" href="/static/styles.css">
   <!-- Include the ZXing library -->
   <script src="{{ url_for('static', filename='index.min.js') }}"></script>
</head>
<body>

<div class="header">
   <div class="navbar">
      <a href="/join_class" class="button">Join Class</a>
      <a href="/profile" class="button">Profile</a>
      <a href="/logout" class="button">Logout</a>
   </div>
</div>

<div class="container">
   <h1>Welcome, {{ user.username }}!</h1>
   <h2>Your Classes:</h2>
   <ul>
      <!-- Loop through classes joined by the student -->
      {% for classroom in joined_classes %}
      <li>
         <a href="{{ url_for('classroom', class_code=classroom.class_code) }}">{{ classroom.class_name
}} - {{ classroom.subject_name }}</a>
         <button class="start-scanning" data-class-name="{{ classroom.class_name }}" data-subject-
name="{{ classroom.subject_name }}">Scan QR Code</button>
      </li>
      {% endfor %}
   </ul>
</div>

<div id="qrModal" class="modal">
   <div class="modal-content">
      <span class="close" id="closeModal">&times;</span>
      <h2>QR Code for Attendance</h2>
      <video id="video" width="300" height="200" style="object-fit: cover;"></video>
   </div>
</div>
```

```
<script>
  document.addEventListener('DOMContentLoaded', function () {
    const startScanningBtns = document.querySelectorAll('.start-scanning');
    const qrModal = document.getElementById('qrModal');
    const cameraFeed = document.getElementById('video');
    let codeReader;

    startScanningBtns.forEach(function (startScanningBtn) {
      startScanningBtn.addEventListener('click', async function () {
        const className = startScanningBtn.getAttribute('data-class-name');
        const subjectName = startScanningBtn.getAttribute('data-subject-name');

        qrModal.style.display = 'block';

        codeReader = new ZXing.BrowserQRCodeReader();

        try {
          const stream = await navigator.mediaDevices.getUserMedia({
            video: { facingMode: { ideal: 'environment' } }
          });
          cameraFeed.srcObject = stream;

          codeReader
            .decodeOnceFromVideoDevice(undefined, 'video')
            .then((result) => {
              if (result) {
                console.log('QR Code result:', result.text);
                processScannedData(result.text, className, subjectName);
                // Close the modal and stop the camera stream
                qrModal.style.display = 'none';
                codeReader.reset();
                if (cameraFeed.srcObject) {
                  const tracks = cameraFeed.srcObject.getTracks();
                  tracks.forEach(track => track.stop());
                }
              } else {
                console.log('No QR code found.');
              }
            })
            .catch((error) => {
              console.error('QR Code scanning error:', error);
              // Close the modal and stop the camera stream on error
              qrModal.style.display = 'none';
```

```
                codeReader.reset();
                if (cameraFeed.srcObject) {
                    const tracks = cameraFeed.srcObject.getTracks();
                    tracks.forEach(track => track.stop());
                }
            });

        } catch (error) {
            console.error('Error starting QR code scanning:', error);
            // Close the modal and stop the camera stream on error
            qrModal.style.display = 'none';
            codeReader.reset();
            if (cameraFeed.srcObject) {
                const tracks = cameraFeed.srcObject.getTracks();
                tracks.forEach(track => track.stop());
            }
        }
    });
});

document.getElementById('closeModal').addEventListener('click', function () {
    qrModal.style.display = 'none';

    // Stop the video stream when closing the modal
    if (codeReader) {
        codeReader.reset();
    }
    if (cameraFeed.srcObject) {
        const tracks = cameraFeed.srcObject.getTracks();
        tracks.forEach(track => track.stop());
    }
});

async function processScannedData(qrCodeData, className, subjectName) {
    const studentUsername = '{{ session.username }}';

    const response = await fetch('/mark_attendance', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            qrCodeData,
            studentUsername,
                className,
                subjectName
            })
        });

        const data = await response.json();
        console.log(data.message);
    }
});
</script>
</body>
</html>
```

- **Teacher_dashboard.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Teacher Dashboard</title>
   <link rel="stylesheet" type="text/css" href="/static/styles.css">
<body>

   <div class="header">
      <div class="navbar">
         <a href="/create_class" class="button">Create Class</a>
         <a href="/profile" class="button">Profile</a>
         <a href="/logout" class="button">Logout</a>
      </div>
   </div>


<div class="container">
   <h1>Welcome, {{ user.username }} !</h1>

   <table>
      <tr>
         <th>Class - Subject</th>
         <th>Class Code</th>
         <th>Action</th>
      </tr>
      {% for classroom in created_classes %}
      <tr>
         <td><a href="{{ url_for('classroom', class_code=classroom.class_code) }}">{{
classroom.class_name }}- {{ classroom.subject_name }}</a></td>
         <td>{{ classroom.class_code }}</td>
         <td>
            <button onclick="generateQRCode('{{ classroom.class_code }}')">Get Attendance</button>
         </td>
      </tr>
      {% endfor %}
   </table>
</div>
```

```
    <script>
        function generateQRCode(classCode) {
            // Make an AJAX request to your Flask route for generating QR code
            fetch(`/generate_qr_code/${classCode}`)
                .then(response => response.blob())
                .then(data => {
                    // Display the QR code in a popup or modal
                    const qrCodeUrl = URL.createObjectURL(data);
                    const popupWindow = window.open('', '_blank');
                    popupWindow.document.write(`<img src="${qrCodeUrl}" alt="QR Code">`);
                    popupWindow.document.title = 'QR Code';
                })
                .catch(error => console.error('Error generating QR code:', error));
        }
    </script>
</body>
</html>
```

- **Create_classroom.html**

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Create Class</title>
   <link rel="stylesheet" type="text/css" href="/static/styles.css">
</head>
<body>

   <div class="header">
      <h1>Welcome, {{ user.username }} (Student)</h1>
      <div class="navbar">
         <a href="/dashboard" class="button">Dashboard</a>
         <a href="/profile" class="button">Profile</a>
         <a href="/logout" class="button">Logout</a>
      </div>
   </div>


<div class="container">
   <h1>Create Classroom</h1>
   <form method="POST">
      <label for="class_name">Class Name:</label>
      <input type="text" name="class_name" required>

      <label for="subject_name">Subject Name:</label>
      <input type="text" name="subject_name" required>

      <button type="submit">Create Classroom</button>
   </form>
</div>
</body>
</html>
```

- **Join_classroom.html**

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Join Class</title>
   <link rel="stylesheet" type="text/css" href="/static/styles.css">
</head>
<body>

   <div class="header">
      <h1>Welcome, {{ user.username }} (Student)</h1>
      <div class="navbar">
         <a href="/dashboard" class="button">Dashboard</a>
         <a href="/profile" class="button">Profile</a>
         <a href="/logout" class="button">Logout</a>
      </div>
   </div>


<div class="container">
   <h1>Join Classroom</h1>
   <form method="POST">
      <label for="class_code">Class Code:</label>
      <input type="text" name="class_code" required>

      <button type="submit">Join Classroom</button>
   </form>
</div>
</body>
</html>
```

- **Classroom.html**

```
<!DOCTYPE html>
<html>
<head>
   <title>Classroom - {{ classroom.class_name }}</title>
   <link rel="stylesheet" type="text/css" href="/static/styles.css">
</head>
<body>

   <div class="header">
      <div class="navbar">
         <a href="/dashboard" class="button">Dashboard</a>
         <a href="/join_class" class="button">Join Class</a>
         <a href="/profile" class="button">Profile</a>
         <a href="/logout" class="button">Logout</a>
      </div>
   </div>

<div class="container">
   <h1>Classroom: {{ classroom.class_name }}</h1>

   <h2>Students:</h2>
   <ul>
     {% for student in students %}
      <li>{{ student }}</li>
   {% endfor %}
   </ul>


   <!-- Add other classroom-specific content here -->
</div>

</body>
</html>
```

CONCLUSION

FEATURE OF STAMS

1. User Authentication: STAMS provides secure user authentication mechanisms to ensure that only authorized users, such as teachers and students, can access the system.

2. Classroom Creation: Teachers can create virtual classrooms and manage class details such as class name, subject, and class code within the system.

3. Student Enrollment: Students can easily enroll in classes by entering the class code provided by their respective teachers.

4. QR Code Generation: Teachers can generate QR codes for each class session to facilitate easy attendance marking.

5. QR Code Scanning: Students can mark their attendance by scanning the QR code displayed during class sessions using their mobile devices.

6. Attendance Tracking: STAMS automatically tracks and records student attendance for each class session, providing teachers with accurate attendance reports.

7. Analytics and Reporting: The system offers analytics and reporting features that enable teachers to analyze attendance data, identify trends, and generate comprehensive reports for each class and student.

8. User Profiles: Users can create and manage their profiles, providing personalized information such as full name, email, and age.

9. Real-time Updates: STAMS provides real-time updates on class enrollment, attendance marking, and system notifications to keep users informed.

10. Ease of Use: The system is designed with a user-friendly interface, making it easy for both teachers and students to navigate and utilize its features effectively.

RECOMMENDATION

1. Continuous Improvement: Encourage continuous improvement of the Student Attendance Management System (STAMS) by soliciting feedback from users, such as teachers and students, and implementing necessary updates and enhancements based on their suggestions.

2. Integration with Educational Platforms: Explore opportunities to integrate STAMS with existing educational platforms and systems used by schools and universities to streamline attendance management processes and enhance overall efficiency.

3. Mobile Application Development: Consider developing a mobile application version of STAMS to provide users with greater flexibility and accessibility, allowing them to manage attendance and access system features on the go.

4. Enhanced Analytics Features: Invest in further development of analytics features within STAMS to provide more in-depth insights into attendance patterns, student engagement, and academic performance, empowering teachers to make data-driven decisions.

5. User Training and Support: Provide comprehensive user training and ongoing technical support to ensure that teachers and students can fully utilize the features of STAMS and address any challenges or issues encountered during system use.

6. Security and Data Privacy: Emphasize the importance of maintaining robust security measures and ensuring data privacy within STAMS to safeguard sensitive information and protect the integrity of the system against potential threats or breaches.

7. Promotion and Adoption: Implement strategies to promote the adoption of STAMS within educational institutions, including conducting awareness campaigns, hosting training workshops, and showcasing the benefits of the system to key stakeholders.

8. Collaboration with Stakeholders: Foster collaboration and partnership with educational institutions, administrators, and technology providers to further refine and expand the capabilities of STAMS, ensuring its relevance and effectiveness in meeting the evolving needs of the education sector.

REFERENCES

REFERENCES

1. Smith, John. "Improving Student Attendance through Technology: A Review of Electronic Attendance Systems." Journal of Educational Technology, vol. 25, no. 2, pp. 123-135, 2020.

2. Johnson, Emily. *Effective Classroom Management: Strategies for Student Engagement* Pearson Education, 2018.

3. "Flask Documentation." Flask Project, https://flask.palletsprojects.com/en/2.1.x/. Accessed March 15, 2024.

4. MongoDB. *MongoDB Documentation* MongoDB Inc., 2023.

5. Personal communication with Dr. Michael Brown, Assistant Professor of Education, University of XYZ. February 28, 2024.

6. "ZXing Documentation." ZXing Project, https://github.com/zxing/zxing. Accessed March 20, 2024.