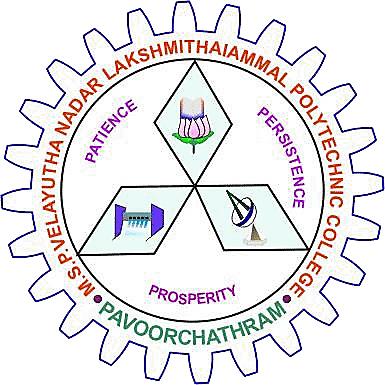
**M.S.P. VELAYUTHA NADAR CHARITY TRUST**

**M.S.P.VELAYUTHA NADAR LAKSHMITHAIAMMAL POLYTECHNIC COLLEGE**

**PAVOORCHATRAM-627808**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**Seamless Web-Based Sports Planner for College Events**

**Project Report: 2024-2025**

**Under the Guidance of,**

**Mrs. S.Jeyalakshmi .B.E.,**

**Submitted by**

**Team Members: Register Number:**

PRIYA DHARSHINI L **-** 23500218

SWAMI SATHYA MURUGAN **-** 23500226

ANIX JEYAVARSHNI C **-** 23590060

JANANI S **-** 23590062

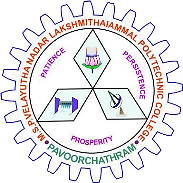
MUKILA S **-** 23590067

**M.S.P. VELAYUTHA NADAR CHARITY TRUST**

**M.S.P.VELAYUTHA NADAR LAKSHMITHAIAMMAL POLYTECHNIC COLLEGE**

**PAVOORCHATRAM-627808**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**Register Number**

Certified that this is a Bonafide Record of Project **Seamless Web-Based Sports Planner for College Events** work was done by \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ of VI **SEMESTER INFORMATION TECHNOLOGY** Department During the year of 2024-2025 for the partial fulfilment of the curriculum of the **DIPLOMA COURSE IN INFORMATION TECHNOLOGY** branch of Directorate of Technical Education, Chennai.

**PROJECT GUIDE HEAD OF THE DEPARTMENT**

Submitted for the Board Practical Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

|  |  |
| --- | --- |
| **Marks Awarded** |  |
| **Total Marks** |  |

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

First and foremost we would like to thank the almighty, who gave us good health and knowledge to do the project successfully. We would never have been able to finish my project without the guidance of my faculty members, help from friends, and support from my family.

We are also happy to thank our Chairman Thiru. **M.S.P.V.P. Rajesh Sankara Kumar** and Correspondent Thiru**. M.S.P.V.K.Lakshmi Anand** for given us a great opportunity to complete our diploma course in reputed institute through this project submission.

We would like to express my deepest gratitude to our honourable principal **Mr.G.Ramesh M.Tech.,** for providing us good laboratory facilities with the advanced configuration systems and without any interruption.

We are delighted to thank **Mr.F.Ashok JayaKumar M.E.,** Head of the Department of Information Technology for his inspiration and guiding us in all our activities of developing this project.

We submit our kind gesture to the project guide **Mrs. S.Jeyalakshmi B.E.,,**

for her guidance and suggestions in executing this project.

We take this opportunity to thank all the Staff Members and Lab Technicians of Information Technology department for their support and suggestion give to our project.

# Last but not least would like to express my gratitude towards my parents and friends for their whole hearted support, help and throughout our course work and project work.



|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE.NO** |
| 1 | ABSTRACT | 3 |
| 2 | INTRODUCTION | 5 |
| 3 | SYSTEM SPECIFICATION | 6 |
| 4 | EXISTING SYSTEM | 8 |
| 5 | PROPOSED SYSTEM | 9 |
| 6 | SOFTWARE DESCRIPTION | 10 |
| 7 | SYSTEM DESIGN AND DEVELOPMENT | 12 |
| 8 | FLOW DIAGRAM | 14 |
| 9 | SAMPLE CODING | 18 |
| 10 | SNAPSHOTS | 47 |
| 11 | CONCLUSION | 56 |
| 12 | REFERENCES | 58 |

**ABSTRACT**



**ABSTRACT OF THE PROJECT**

Seamless Web-Based Sports Planner for College Events is an advanced digital platform a designed to streamline and optimize the planning, coordination, and execution of sports events within a college environment. This integrated system offers a comprehensive solution for managing all facets of sports events, from initial scheduling and participant registration to final results and prize distribution. The primary goal of this project is to enhance the efficiency and effectiveness of sports event management by providing a centralized platform that integrates various administrative and organizational tasks. By offering tailored functionalities for students, Physical Directors, and administrators, the platform aims to facilitate smooth event operations, improve communication, and ensure a seamless experience for all stakeholders involved.

**INTRODUCTION**

* Sports events play a crucial role in fostering teamwork, discipline, and healthy competition in college environments. However, managing these events manually can be time-consuming and prone to errors. The **Seamless Web-Based Sports Planner for College Events** is designed to **digitize and streamline** the entire event management process. This platform enables **efficient scheduling, participant registration, real-time result tracking, and prize distribution** through a centralized system. By offering tailored features for **students, Physical Directors, and administrators**, it ensures smooth event coordination, enhances communication, and reduces administrative workload.



**SYSTEM SPECIFICATION**



**SOFTWARE REQUIREMENTS**

Front-end : HTML & CSS

Framework : Flask

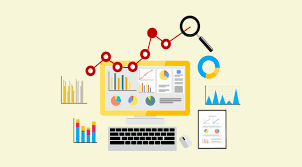
Back-end : Python 3.8

DataBase : DB Browser**(** Sqlite )

Application : Pycharm Community

**EXISTING SYSTEM**

* **Paper-Based Registration**: Time-consuming and prone to errors.
* **Manual Scheduling**: Leads to mismanagement and clashes.
* **Offline Result Compilation**: Difficult to track performance and generate leaderboards.
* **Lack of Notifications**: No real-time updates for participants.
* **Limited Accessibility**: Information is not easily accessible to students and staff.



**PROPOSED SYSTEM**

* **User-Friendly Dashboards**: Role-based access for admins, students, and staff.
* **Online Registration:** Students and staff can register individually or as teams.
* **Email & Notification System**: Send event updates and registration confirmations.



**SOFTWARE DESCRIPTION**



**Pycharm professional:**

* PyCharm **Professional Edition** is a powerful **Integrated Development Environment (IDE)** developed by **JetBrains** for Python developers.
* It offers advanced tools for **web development, database management, and remote development**, making it ideal for professional use.
* The IDE provides an **intelligent coding environment** with smart completion, real-time error detection, and automated refactoring.
* It supports **frameworks like Django and Flask**, enabling seamless full-stack development.
* **Database integration** allows direct access to SQL databases within the IDE.
* Built-in **debugging and testing tools** help optimize code performance efficiently.
* Remote development through **SSH, Docker, and WSL** enhances flexibility.

**Features of pycharm professional:**

* PyCharm Professional Edition is a powerful integrated development environment (IDE) designed for advanced Python development.
* It offers built-in support for web frameworks like Django, Flask, and FastAPI, along with database integration, allowing direct connections to SQL databases such as MySQL and PostgreSQL.
* The IDE provides advanced debugging, testing, and profiling tools, making it easier to analyze and optimize code.
* It also supports remote development through SSH, Docker, and WSL, enabling seamless collaboration.
* With AI-powered code assistance, smart refactoring, and real-time error detection, PyCharm Professional enhances productivity and is ideal for full-stack development, data science, and enterprise applications.

**About DB Browser (Sqlite):**



* It was originally created by **Maurice Soft** in **2003** and later maintained by the open-source community. It supports **table creation, data import/export (CSV, SQL, JSON), and direct query execution.** The software is cross-platform, available for **Windows, macOS, and Linux**.
* It allows users to create, edit, browse, and manage SQLite databases without needing to write SQL commands manually.
* With features like table creation, data import/export, and query execution, it simplifies database management for developers and non-technical users.
* It is widely used in software development, data analysis, and embedded applications, making SQLite database management more accessible and efficient.

**SYSTEM DESIGN AND DEVELOPMENT**

* Home module
* Registration module
* Pd Login module
* Staff Login module
* Game Description module
* Game Rounds module
* Winner/Runner module
* Certificate module

**Home Module:**

* The Home Module is the central hub of a web-based or software application, providing users with a starting point and easy navigation to key features. It serves as the main dashboard where users can access various sections of the system.

**Registration Module:**

* The Registration Module is a crucial component of any system that allows users to sign up and gain access to the platform. It ensures secure and structured user onboarding by collecting necessary details and managing user authentication.

**PD login Module:**

* The PD Login Module is designed specifically for the Physical Director (Admin) to securely access and manage the sports event system. It provides authentication and ensures that only authorized personnel can perform administrative tasks.

**Staff Login Module:**

* The **Staff Login Module** allows faculty members (staff) to securely access the sports event system. It ensures role-based authentication, enabling staff to participate in and manage specific activities within the platform.

**Game Description Module:**

* The Game Description Module provides detailed information about each sports event, helping participants understand the rules, schedules, and guidelines. It acts as an informational hub for students, staff, and administrators.

**Game Rounds Module:**

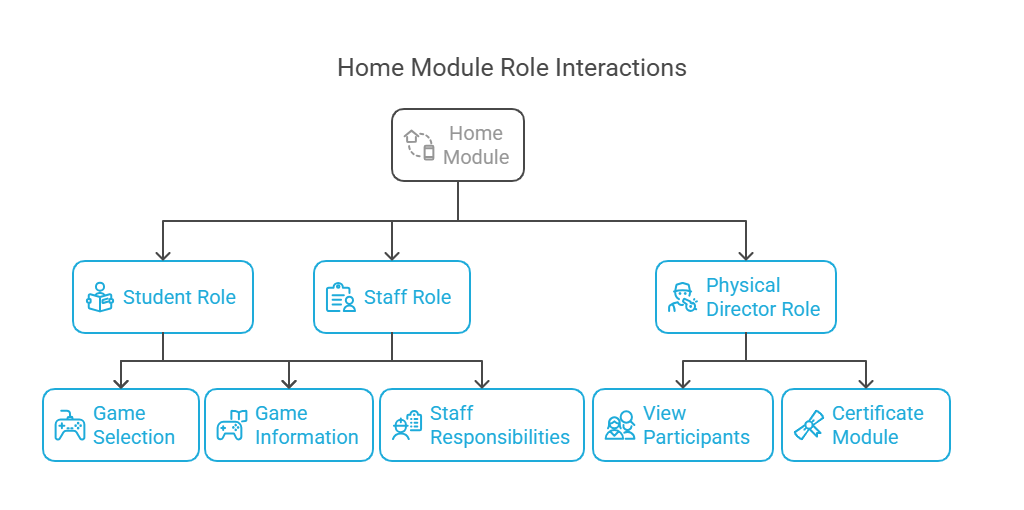
* The **Game Rounds Module** is responsible for managing and displaying different stages of a sports event, ensuring smooth progression from the initial rounds to the finals. It helps organizers, participants, and spectators track the competition structure.

**Winner/Runner Module:**

* The **Winner/Runner Module** is responsible for recording, displaying, and managing the final results of sports events. It ensures a transparent and structured way to recognize the **winners and runners-up** of each competition.

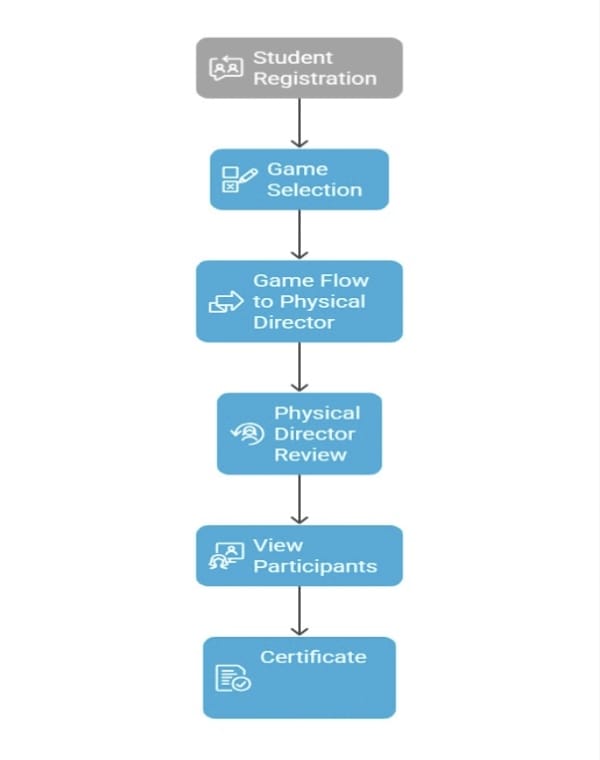
**Certificate Module:**

* The **Certificate Module** is responsible for generating and distributing **digital or printable certificates** to participants, winners, and runners-up after the completion of sports events. It ensures proper recognition of achievements in an organized and automated manner.

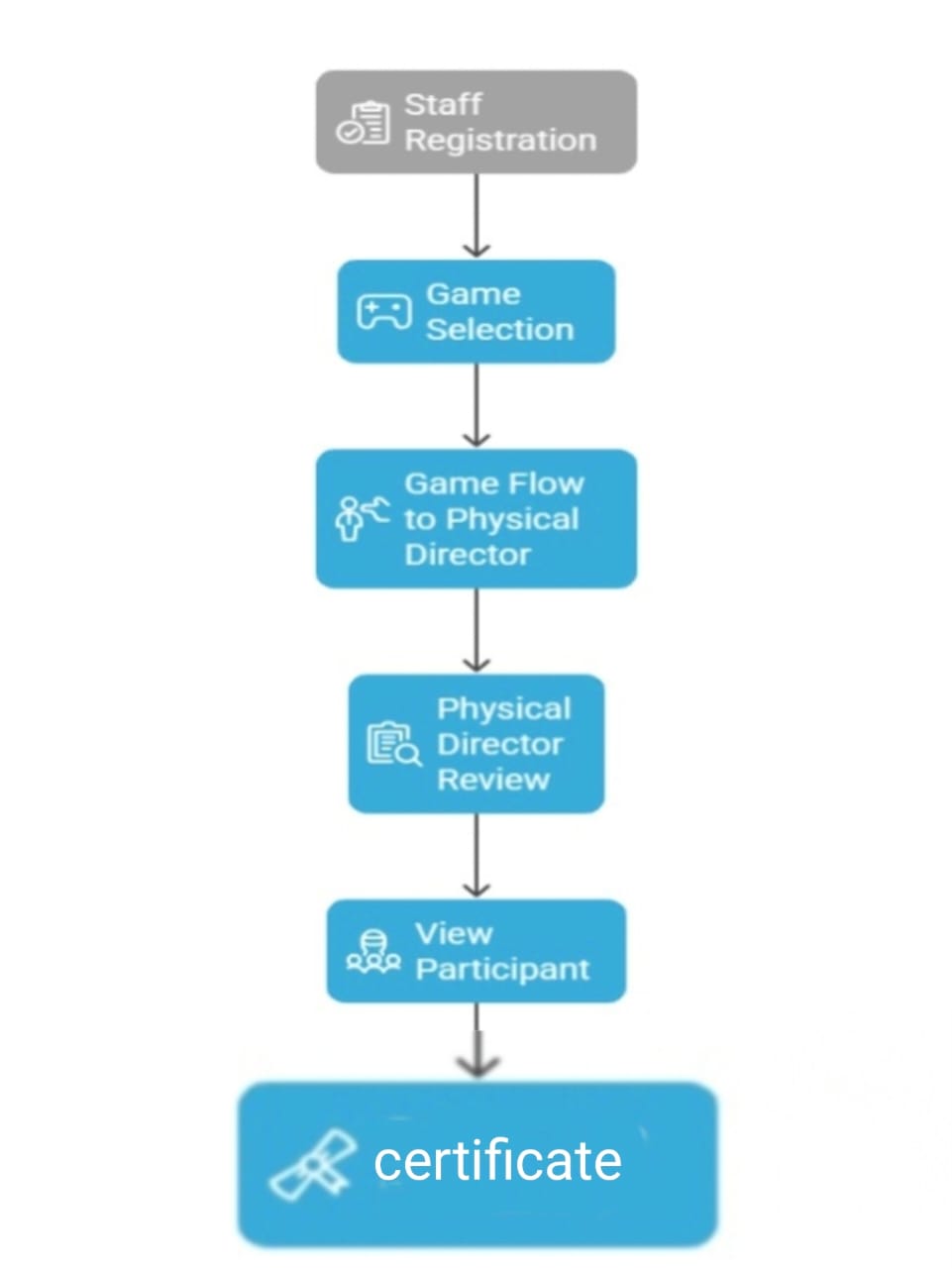
**FLOW DIAGRAM**

**DATA FLOW DIAGRAM:**

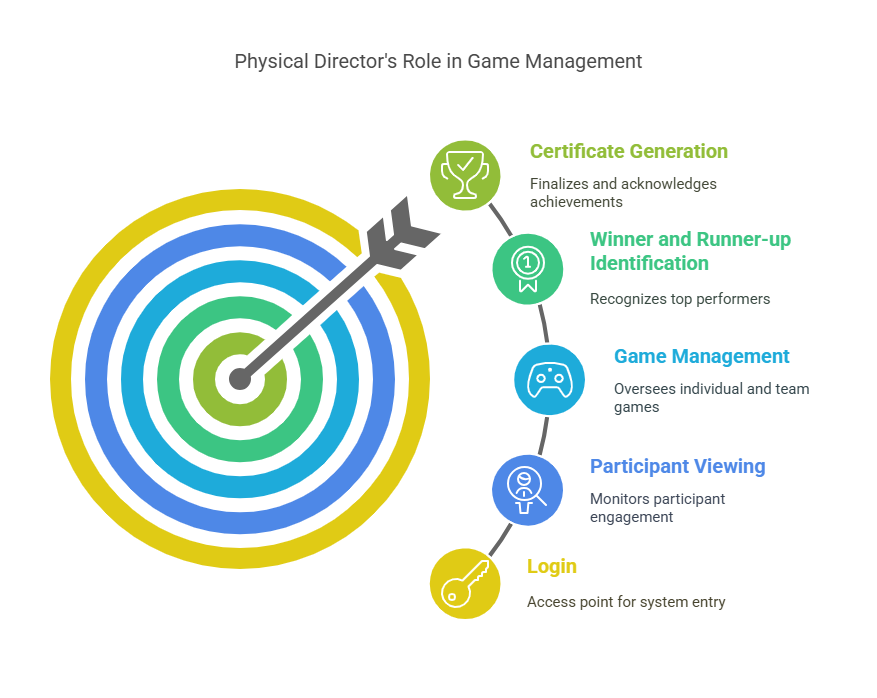
**1.Student Game Registration Process:**

****

**2.Staff Game Registration Process:**

****

**3.Physical director Game Registration Process:**



**SAMPLE CODING**

****

**App.py**

from flask import Flask, render\_template, request,url\_for,redirect,session,flash  
import sqlite3  
from werkzeug.security import generate\_password\_hash,check\_password\_hash  
import os  
  
app = Flask(\_\_name\_\_)  
app.config['SECRET\_KEY'] =os.urandom(24)  
  
*# Create or connect to an SQLite database*def init\_db():  
 con = sqlite3.connect("sports25.db")  
 print("sports Database opened successfully")  
  
 con.execute('CREATE TABLE IF NOT EXISTS student\_reg (ID INTEGER PRIMARY KEY AUTOINCREMENT,Name TEXT NOT NULL,\  
 Department TEXT NOT NULL,Reg\_No INTEGER UNIQUE NOT NULL,Year TEXT NOT NULL,Email TEXT UNIQUE NOT NULL,\  
 Password TEXT NOT NULL,Ind\_game TEXT NOT NULL,Group\_game TEXT NOT NULL)')  
  
 print("student\_reg Table created successfully")  
  
 con.execute('CREATE TABLE IF NOT EXISTS staff\_reg (ID INTEGER PRIMARY KEY AUTOINCREMENT,Name TEXT NOT NULL,\  
 Department TEXT NOT NULL,Email TEXT UNIQUE NOT NULL,Password TEXT NOT NULL,Ind\_game TEXT NOT NULL,Group\_game TEXT NOT NULL)')  
  
  
 print("staff\_reg Table created successfully")  
  
 con.execute('CREATE TABLE IF NOT EXISTS pd\_signup(ID INTEGER PRIMARY KEY AUTOINCREMENT,Username TEXT UNIQUE NOT NULL,\

Password TEXT NOT NULL)')  
  
 print("pd\_signup Table created successfully")  
  
 *# Add a sample user with a hashed password* Username = 'testuser'  
 Password = 'password123'  
 hash\_pwd = generate\_password\_hash(Password)  
  
 con.execute("INSERT OR IGNORE INTO pd\_signup (Username, Password) VALUES (?, ?)", (Username, hash\_pwd))  
 con.commit()  
 con.close()  
  
  
def check\_user\_credentials(uname,pwd):  
 with sqlite3.connect("sports25.db") as check1:  
 cursor = check1.cursor()  
 *#Retrieve the user data based on username* cursor.execute("SELECT \* FROM pd\_signup WHERE Username = ?", (uname,))  
 user = cursor.fetchone()  
 if user:  
 stored\_password = user[2]  
 return check\_password\_hash(stored\_password, pwd)  
 return False  
  
  
  
@app.before\_request  
def setup():  
 init\_db()  
  
@app.route('/')  
@app.route('/home')

def index():  
 return render\_template('index.html')  
  
@app.route('/thankyou')  
def thankyou():  
 return render\_template('thankyou.html')  
  
@app.route('/success')  
def success():  
 return render\_template('success.html')  
  
@app.route('/failure')  
def failure():  
 return render\_template('failure.html')  
@app.route('/stud\_reg1')  
def stud\_reg1():  
 return render\_template('stud\_reg1.html')  
  
@app.route("/stud\_reg1\_info",methods = ["POST","GET"])  
def stud\_reg1\_info():  
 if request.method == "POST":  
 Name = request.form['name']  
 Department = request.form['dept']  
 Reg\_No = request.form['reg\_no']  
 Year = request.form['year']  
 Email = request.form['email']  
 Password = request.form['pwd1']  
  
 session['Name'] = Name  
 session['Department'] = Department  
 session['Reg\_No'] = Reg\_No  
 session['Year'] = Year  
 session['Email'] = Email  
 session['Password'] = Password

*# Redirect to step 2 for choosing games* return redirect(url\_for('stud\_reg2'))  
 return render\_template('stud\_reg1.html')  
  
*# Step 2: Register Games*@app.route('/stud\_reg2')  
def stud\_reg2():  
 return render\_template('stud\_reg2.html')  
@app.route('/stud\_reg2\_info', methods=['GET', 'POST'])  
def stud\_reg2\_info():  
 msg = "msg"  
 if request.method == 'POST':  
 try:  
 *# Handle multiple checkboxes* Ind\_game = request.form.getlist('i-opt')  
 *# get selected checkbox values* Ind\_game\_str = ','.join(Ind\_game)  
 *# convert list to comma-separated string* Group\_game = request.form.getlist('g-opt')  
 *#get selected checkbox values* Group\_game\_str = ','.join(Group\_game)  
  
 *# Get details from stud\_reg1 session* Name=session.get('Name')  
 Department=session.get('Department')  
 Reg\_No=session.get('Reg\_No')  
 Year=session.get('Year')  
 Email=session.get('Email')  
 Password=session.get('Password')  
  
 with sqlite3.connect("sports25.db") as reg1:  
 cur = reg1.cursor()

cur.execute("INSERT INTO student\_reg(Name,Department,Reg\_No,Year,Email,Password,Ind\_game,Group\_game)VALUES (?,?,?,?,?,?,?,?)",\  
 (Name, Department, Reg\_No, Year, Email, Password,Ind\_game\_str, Group\_game\_str))  
 reg1.commit()  
 flash("Registration successful!", "success...")  
 msg = "Dear Student,Your Sports Day Registration Completed Successfully..."  
  
 except:  
 reg1.rollback()  
 flash("Registration Failed!", "Try Again...")  
 msg = "Sorry,Your Sports Day Registration Failed...Try again..."  
  
 finally:  
 reg1.close()  
 return render\_template("success.html", msg=msg)  
  
 return render\_template('stud\_reg2.html')  
  
  
  
@app.route('/staff\_reg1')  
def staff\_reg1():  
 return render\_template('staff\_reg1.html')  
  
@app.route("/staff\_reg1\_info",methods = ["POST","GET"])  
def staff\_reg1\_info():  
 if request.method == "POST":  
  
 Name = request.form['name']  
 Department = request.form['dept']  
 Email = request.form['email']  
 Password = request.form['pwd1']

session['Name'] = Name  
 session['Department'] = Department  
 session['Email'] = Email  
 session['Password'] = Password  
 *# Redirect to step 2 for choosing games* return redirect(url\_for('staff\_reg2'))  
 return render\_template('staff\_reg1.html')  
  
@app.route('/staff\_reg2')  
def staff\_reg2():  
 return render\_template('staff\_reg2.html')  
  
@app.route("/staff\_reg2\_info",methods=['GET', 'POST'])  
def staff\_reg2\_info():  
 msg = "msg"  
 if request.method == "POST":  
 try:  
 *# Handle multiple checkboxes* Ind\_game = request.form.getlist('i-opt')  
 *# get selected checkbox values* Ind\_game\_str = ','.join(Ind\_game)  
 *# convert list to comma-separated string* Group\_game = request.form.getlist('g-opt')  
 Group\_game\_str = ','.join(Group\_game)  
  
 *# Get details from staff\_reg1 session* Name=session.get('Name')  
 Department=session.get('Department')  
 Email=session.get('Email')  
 Password=session.get('Password')  
  
  
 with sqlite3.connect("sports25.db") as reg2:

cur = reg2.cursor()  
  
 cur.execute("INSERT INTO staff\_reg(Name,Department,Email,Password,Ind\_game,Group\_game)VALUES (?,?,?,?,?,?)",\  
 (Name,Department,Email,Password,Ind\_game\_str,Group\_game\_str))  
 reg2.commit()  
 msg = ("Dear Staff,Your Sports Day Registration Completed Successfully...")  
 except:  
 reg2.rollback()  
 msg = "Sorry,Your Sports Day Registration Failed...Try again..."  
 finally:  
 reg2.close()  
 return render\_template("success.html",msg = msg)  
 return render\_template('staff\_reg2.html')  
  
@app.route('/pd\_gen')  
def pd\_gen():  
 return render\_template('pd\_gen.html')  
  
@app.route('/pd\_signup')  
def pd\_signup():  
 return render\_template('pd\_signup.html')  
  
@app.route('/pd\_login')  
def pd\_login():  
 return render\_template('pd\_login.html')  
  
@app.route("/pd\_signup\_info",methods = ["POST","GET"])  
def pd\_signup\_info():  
 msg = "msg"  
 if request.method == "POST":  
 try:  
 Username = request.form['uname']

Password = request.form['pwd']  
 hash\_pwd = generate\_password\_hash(Password)  
 with sqlite3.connect("sports25.db") as reg3:  
 cur = reg3.cursor()  
 cur.execute("INSERT INTO pd\_signup(Username,Password) VALUES (?,?)",  
 (Username,hash\_pwd))  
 reg3.commit()  
 msg = "Hello Physical Director,Your Signup Process Completed Successfully..."  
 except:  
 reg3.rollback()  
 msg = "Sorry,Your Signup Process Failed...Try Again..."  
 finally:  
 reg3.close()  
 return render\_template("success.html",msg = msg)  
  
@app.route('/pd\_login\_info', methods=['POST'])  
def pd\_login\_info():  
 uname=request.form['uname']  
 pwd=request.form['pwd']  
 if check\_user\_credentials(uname,pwd):  
 session['username'] = uname *# Store the username in session* flash('Login successful!', 'success')  
 return redirect(url\_for('pd\_view'))  
 else:  
 flash('Invalid credentials, please try again.', 'danger')  
 return redirect(url\_for('pd\_gen'))  
  
  
@app.route('/pd\_view')  
def pd\_view():  
 return render\_template('pd\_view.html')  
  
@app.route('/pd\_dashboard')  
def pd\_dashboard():

if 'username' not in session:  
 return redirect(url\_for('pd\_gen'))  
 *#print(f"Welcome, {session['username']}! Please wait..you are being redirected to view the dashboard.")* else:  
 return redirect(url\_for('pd\_view'))  
  
@app.route('/pd\_logout')  
def pd\_logout():  
 session.pop('username', None) *# Remove user from session* flash('You have been logged out.', 'info')  
 return redirect(url\_for('pd\_gen'))  
  
  
@app.route('/certificate')  
def certificate():  
 return render\_template('certificate.html')  
  
  
@app.route("/view\_student")  
def view\_student():  
 view1 = sqlite3.connect("sports25.db")  
 view1.row\_factory = sqlite3.Row  
 cur = view1.cursor()  
 cur.execute("select \* from student\_reg")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg")  
 row\_count = cur.fetchone()[0]  
 return render\_template("view\_student.html",rows = rows,row\_count=row\_count)  
  
  
@app.route("/view\_staff")  
def view\_staff():  
 view2 = sqlite3.connect("sports25.db")

view2.row\_factory = sqlite3.Row  
 cur = view2.cursor()  
 cur.execute("select \* from staff\_reg")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg")  
 row\_count = cur.fetchone()[0]  
 return render\_template("view\_staff.html",rows = rows,row\_count=row\_count)  
  
@app.route("/carrom\_stuview")  
def carrom\_stuview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 *#query="select \* from student\_reg where Ind\_game like ?"  
 #filter="%" + carrom + "%"  
 #filter1 = " %,carrom" or "carrom,%" or "%,carrom,%"  
 #cur.execute(query,(filter,))* cur.execute("select \* from student\_reg where Ind\_game like '%Carrom%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg where Ind\_game like '%Carrom%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("carrom\_stuview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/carrom\_staffview")  
def carrom\_staffview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from staff\_reg where Ind\_game like '%Carrom%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg where Ind\_game like '%Carrom%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("carrom\_staffview.html",rows = rows,row\_count=row\_count)

@app.route("/chess\_stuview")  
def chess\_stuview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from student\_reg where Ind\_game like '%Chess%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg where Ind\_game like '%Chess%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("chess\_stuview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/chess\_staffview")  
def chess\_staffview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from staff\_reg where Ind\_game like '%Chess%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg where Ind\_game like '%Chess%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("chess\_staffview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/discuss\_stuview")  
def discuss\_stuview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from student\_reg where Ind\_game like '%Discuss Throw%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg where Ind\_game like '%Discuss Throw%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("discuss\_stuview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/discuss\_staffview")

def discuss\_staffview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from staff\_reg where Ind\_game like '%Discuss Throw%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg where Ind\_game like '%Discuss Throw%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("discuss\_staffview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/shotput\_stuview")  
def shotput\_stuview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from student\_reg where Ind\_game like '%Shot Put%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg where Ind\_game like '%Shot Put%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("shotput\_stuview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/shotput\_staffview")  
def shotput\_staffview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from staff\_reg where Ind\_game like '%Shot Put%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg where Ind\_game like '%Shot Put%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("shotput\_staffview.html",rows = rows,row\_count=row\_count)  
  
  
@app.route("/running\_stuview")

def running\_stuview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from student\_reg where Ind\_game like '%Running(100m)%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg where Ind\_game like '%Running(100m)%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("running\_stuview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/running\_staffview")  
def running\_staffview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from staff\_reg where Ind\_game like '%Running(100m)%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg where Ind\_game like '%Running(100m)%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("running\_staffview.html",rows = rows,row\_count=row\_count)  
  
  
@app.route("/throw\_stuview")  
def throw\_stuview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from student\_reg where Group\_game like '%Throw Ball%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg where Group\_game like '%Throw Ball%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("throw\_stuview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/throw\_staffview")

def throw\_staffview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from staff\_reg where Group\_game like '%Throw Ball%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg where Group\_game like '%Throw Ball%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("throw\_staffview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/foot\_stuview")  
def foot\_stuview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from student\_reg where Group\_game like '%Foot Ball%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg where Group\_game like '%Foot Ball%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("foot\_stuview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/foot\_staffview")  
def foot\_staffview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from staff\_reg where Group\_game like '%Foot Ball%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg where Group\_game like '%Foot Ball%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("foot\_staffview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/basket\_stuview")  
def basket\_stuview():

con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from student\_reg where Group\_game like '%Basket Ball%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg where Group\_game like '%Basket Ball%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("basket\_stuview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/basket\_staffview")  
def basket\_staffview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from staff\_reg where Group\_game like '%Basket Ball%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg where Group\_game like '%Basket Ball%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("basket\_staffview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/cricket\_stuview")  
def cricket\_stuview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from student\_reg where Group\_game like '%Cricket%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg where Group\_game like '%Cricket%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("cricket\_stuview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/cricket\_staffview")  
def cricket\_staffview():  
 con2 = sqlite3.connect("sports25.db")

con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from staff\_reg where Group\_game like '%Cricket%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg where Group\_game like '%Cricket%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("cricket\_staffview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/volley\_stuview")  
def volley\_stuview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from student\_reg where Group\_game like '%Volley Ball%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from student\_reg where Group\_game like '%Volley Ball%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("volley\_stuview.html",rows = rows,row\_count=row\_count)  
  
@app.route("/volley\_staffview")  
def volley\_staffview():  
 con2 = sqlite3.connect("sports25.db")  
 con2.row\_factory = sqlite3.Row  
 cur =con2.cursor()  
 cur.execute("select \* from staff\_reg where Group\_game like '%Volley Ball%' ")  
 rows = cur.fetchall()  
 cur.execute("select count(\*) from staff\_reg where Group\_game like '%Volley Ball%' ")  
 row\_count = cur.fetchone()[0]  
 return render\_template("volley\_staffview.html",rows = rows,row\_count=row\_count)  
  
  
@app.route('/semi\_cricket\_selected', methods=['POST'])  
def semi\_cricket\_selected():  
 selected\_ids = request.form.getlist('selected\_ids')  
  
 if selected\_ids:  
 try:  
 con = sqlite3.connect("sports25.db")  
 con.row\_factory = sqlite3.Row  
 cur = con.cursor()  
  
 selected\_ids = [int(i) for i in selected\_ids]  
 placeholders = ','.join('?' for \_ in selected\_ids)  
  
 query = f"""  
 SELECT \* FROM student\_reg  
 WHERE id IN ({placeholders})  
 AND Group\_game LIKE '%Cricket%'  
 """  
  
 cur.execute(query, selected\_ids)  
 selected\_participants = cur.fetchall()  
 con.close()  
  
 *# Only keep "Cricket" in the Group\_game field when rendering* clean\_participants = []  
 for participant in selected\_participants:  
 temp = dict(participant)  
 temp['Group\_game'] = 'Cricket' *# Force to show only Cricket* clean\_participants.append(temp)  
  
 return render\_template('cri\_semi\_selected.html', selected\_participants=clean\_participants)  
  
 except Exception as e:  
 return f"Error occurred: {e}"  
 else:  
 return "No participants selected."  
  
  
@app.route('/semi\_throwball\_selected', methods=['GET', 'POST'])  
def semi\_throwball\_selected():  
 try:  
 con = sqlite3.connect("sports25.db")  
 con.row\_factory = sqlite3.Row  
 cur = con.cursor()  
  
 error\_message = None *# For displaying "No participants selected" in the template* if request.method == 'POST':  
 selected\_ids = request.form.getlist('selected\_ids')  
  
 if selected\_ids:  
 selected\_ids = [int(i) for i in selected\_ids]  
 placeholders = ','.join('?' for \_ in selected\_ids)  
  
 query = f"""  
 SELECT \* FROM student\_reg  
 WHERE id IN ({placeholders})  
 AND Group\_game LIKE '%Throw Ball%'  
 """  
 cur.execute(query, selected\_ids)  
 else:  
 *# User clicked submit but didn't select anyone* error\_message = "⚠ Please select at least one participant."  
  
 *# Load default list again* query = """  
 SELECT \* FROM student\_reg  
 WHERE Group\_game LIKE '%Throw Ball%'  
 """  
 cur.execute(query)  
 else:  
 *# GET request — show all throw ball participants*

query = """  
 SELECT \* FROM student\_reg  
 WHERE Group\_game LIKE '%Throw Ball%'  
 """  
 cur.execute(query)  
  
 selected\_participants = cur.fetchall()  
 con.close()  
  
 *# Normalize group name* clean\_participants = []  
 for participant in selected\_participants:  
 temp = dict(participant)  
 temp['Group\_game'] = 'Throw Ball'  
 clean\_participants.append(temp)  
  
 return render\_template(  
 'throw\_semi\_selected.html',  
 selected\_participants=clean\_participants,  
 row\_count=len(clean\_participants),  
 error\_message=error\_message  
 )  
  
 except Exception as e:  
 return f"Error occurred: {e}"  
"""  
connect = sqlite3.connect('sports.db')  
connect.execute(  
 'CREATE TABLE IF NOT EXISTS student\_reg (Name TEXT,Department TEXT,Reg\_No INT PRIMARY KEY, Year TEXT,Email TEXT, Password TEXT, Conf\_Password TEXT)')  
@app.route('/stud\_reg', methods=['GET', 'POST'])  
def stud\_reg():

if request.method == 'POST':  
 Name = request.form['name']  
 Department=request.form['dept']  
 Reg\_No=request.form['reg\_no']  
 Year = request.form['year']  
 Email = request.form['email']  
 Password = request.form['pwd1']  
 Conf\_Password= request.form['pwd2']  
  
  
 with sqlite3.connect("sports.db") as users:  
 cursor = users.cursor()  
 cursor.execute("INSERT INTO student\_reg(Name,Department,Reg\_No,Year,Email,Password,Conf\_Password) VALUES (?,?,?,?,?,?,?)",  
 (Name,Department,Reg\_No,Year,Email,Password,Conf\_Password))  
 users.commit()  
 return render\_template("index.html")  
 else:  
 return render\_template("stud\_reg1.html")  
@app.route('/participants')  
def participants():  
 connect = sqlite3.connect('database.db')  
 cursor = connect.cursor()  
 cursor.execute('SELECT \* FROM PARTICIPANTS')  
  
 data = cursor.fetchall()  
 return render\_template("participants.html", data=data)  
"""  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

**Registeration.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Student Registration Form</title>  
 <style>  
 body {  
 font-family: Arial, sans-serif;  
 background: linear-gradient(to right, #fbc2eb, #a6c1ee); */\* Pastel Gradient \*/* display: flex;  
 justify-content: center;  
 align-items: center;  
 height: 100vh;  
 margin: 0;  
 }  
 .container {  
 background: rgba(255, 255, 255, 0.3);  
 backdrop-filter: blur(10px);  
 padding: 30px;  
 border-radius: 15px;  
 box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);  
 width: 70%;  
 max-width: 900px;  
 text-align: center;  
 color: #333;  
 display: flex;  
 flex-wrap: wrap;  
 justify-content: space-between;  
 }  
 h2 {  
 width: 100%;

color: #444;  
 margin-bottom: 15px;  
 }  
 .form-group {  
 width: 48%;  
 margin-bottom: 15px;  
 text-align: left;  
 }  
 label {  
 font-weight: bold;  
 display: block;  
 margin-bottom: 5px;  
 }  
 input, select {  
 width: 100%;  
 padding: 10px;  
 border: none;  
 border-radius: 8px;  
 outline: none;  
 background: rgba(255, 255, 255, 0.5);  
 color: #333;  
 font-size: 14px;  
 }  
 input::placeholder {  
 color: #666;  
 }  
 .full-width {  
 width: 100%;  
 }  
 button {  
 width: 100%;  
 background: #ffafbd;  
 color: white;  
 padding: 12px;

border: none;  
 border-radius: 8px;  
 cursor: pointer;  
 font-size: 16px;  
 margin-top: 10px;  
 transition: 0.3s;  
 }  
 button:hover {  
 background: #ff7eb3;  
 }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <h2>Student Registration Form</h2>  
 <form action = "/stud\_reg1\_info" method="post" style="width: 100%; display: flex; flex-wrap: wrap; justify-content: space-between;">  
 <div class="form-group">  
 <label for="name">Name:</label>  
 <input type="text" id="name" name="name" placeholder="Enter your name" required>  
 </div>  
  
 <div class="form-group">  
 <label for="dept">Department:</label>  
 <select id="dept" name="dept" required>  
 <option value="">Select Department</option>  
 <option value="Auto">Automobile Engineering</option>  
 <option value="Mech">Mechanical Engineering</option>  
 <option value="ECE">Electronics and Communication Engineering</option>  
 <option value="EEE">Electrical and Electronic Engineering</option>  
 <option value="CSE">Computer Engineering</option>  
 <option value="IT">Information Technology</option>  
 <option value="Civil">Civil Engineering</option>

</select>  
 </div>  
  
 <div class="form-group">  
 <label for="reg\_no">Register Number:</label>  
 <input type="text" id="reg\_no" name="reg\_no" placeholder="Enter register number" required>  
 </div>  
  
 <div class="form-group">  
 <label for="year">Year:</label>  
 <select id="year" name="year" required>  
 <option value="">Select Year</option>  
 <option value="I">I</option>  
 <option value="II ">II</option>  
 <option value="III">III</option>  
 </select>  
 </div>  
  
 <div class="form-group">  
 <label for="email">Email:</label>  
 <input type="email" id="email" name="email" placeholder="Enter email" required>  
 </div>  
  
 <div class="form-group">  
 <label for="password">Password:</label>  
 <input type="password" id="password" name="pwd1" placeholder="Enter password" required>  
 </div>  
  
 <div class="form-group full-width">  
 <button type="submit">Next</button>  
 </div> </form> </div> </body> </html>  
  
**PDview.html:**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Physical Director Page</title>  
 <style>  
 @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@500&display=swap');  
  
 body {  
 font-family: 'Poppins', sans-serif;  
 background: linear-gradient(to right, #a1c4fd, #c2e9fb); */\* Pastel Gradient \*/* display: flex;  
 justify-content: center;  
 align-items: center;  
 height: 200vh;  
 margin: 0;  
 }  
  
 .container {  
 background: rgba(255, 255, 255, 0.2);  
 backdrop-filter: blur(10px);  
 padding: 30px;  
 border-radius: 15px;  
 box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);  
 width: 1000px;  
 height:160vh;  
 text-align: center;  
 }

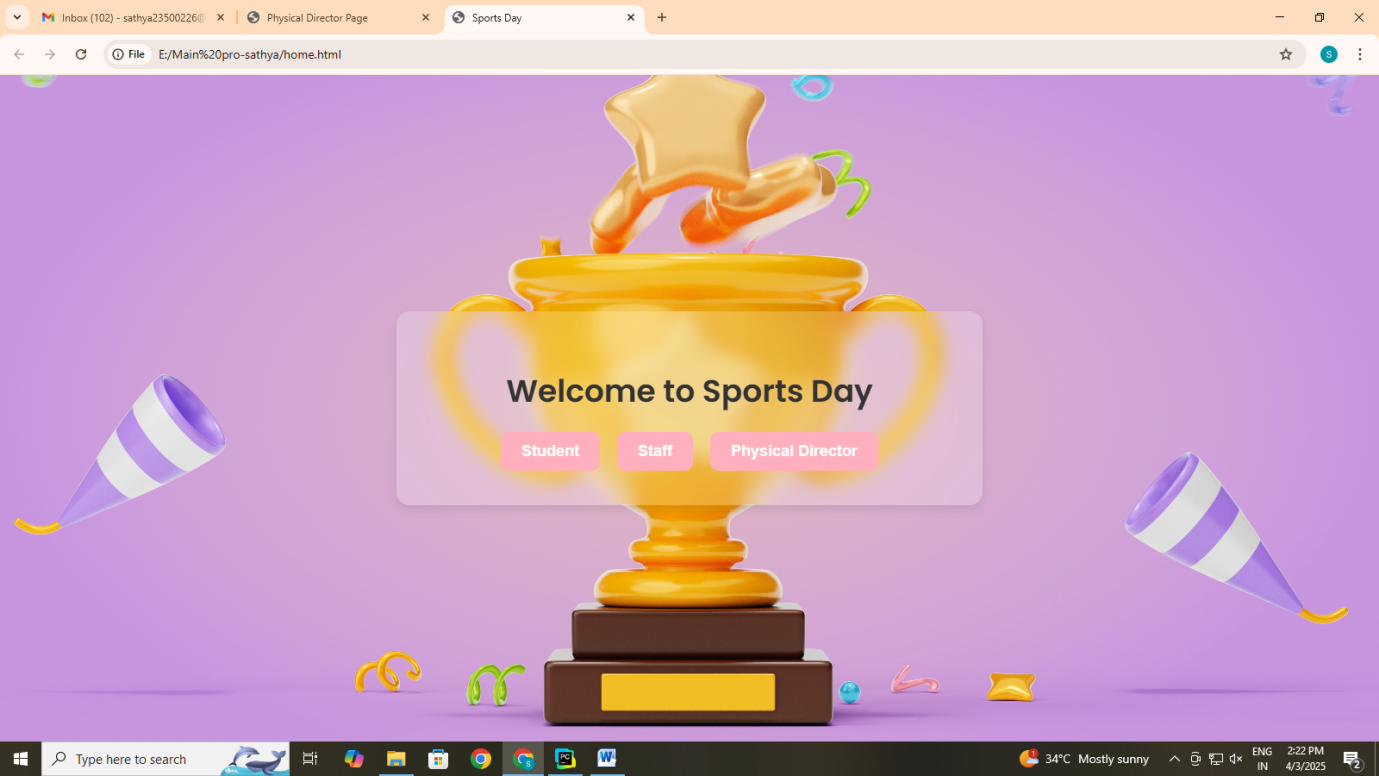
h2 {  
 font-size: 28px;  
 color: #333;  
 margin-bottom: 20px;  
 }  
  
 .button-container {  
 display: grid;  
 grid-template-columns: 1fr 1fr;  
 gap: 15px;  
 margin-bottom: 20px;  
 }  
  
 .btn {  
 width: 75%;  
 background: #ff758c;  
 color: white;  
 padding: 15px;  
 border: none;  
 border-radius: 10px;  
 cursor: pointer;  
 font-size: 16px;  
 font-weight: bold;  
 transition: 0.3s;  
 }  
  
 .btn:hover {  
 background: #ff5e78;  
 transform: scale(1.05);  
 }  
 .center-btn {  
 width: 80%;  
 margin-top: 15px;

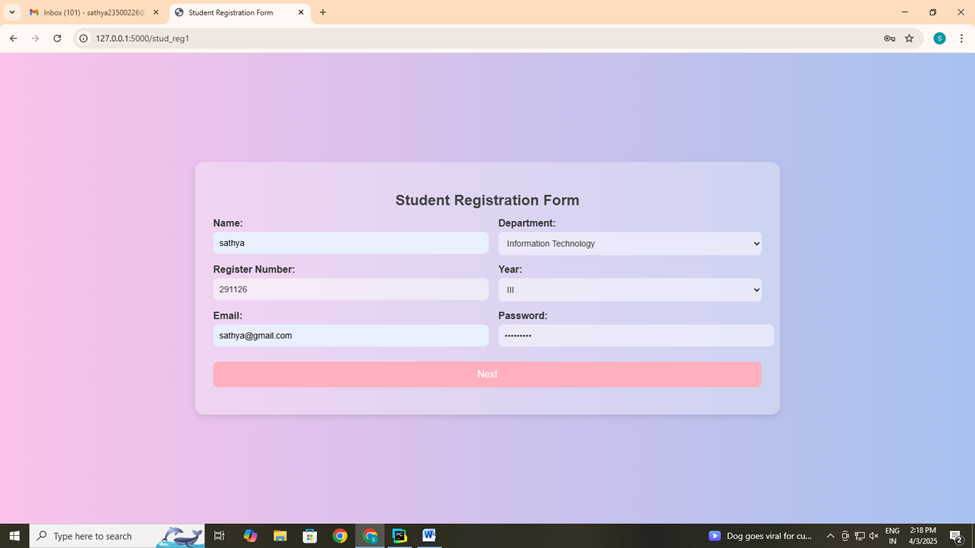
}  
 </style>  
</head>  
<body>  
 <div class="container">  
 <h2>Welcome to Physical Director Page</h2>  
 <div class="button-container">  
 <button class="btn" onclick="window.location.href='/view\_student'">Overall Student Participants</button>  
 <button class="btn" onclick="window.location.href='/view\_staff'">Overall Staff Participants</button>  
  
 <button class="btn" onclick="window.location.href='/carrom\_stuview'">Carrom(stu) Participants</button>  
 <button class="btn" onclick="window.location.href='/carrom\_staffview'">Carrom(staff) Participants</button>  
  
 <button class="btn" onclick="window.location.href='/chess\_stuview'">Chess(stu) Participants</button>  
 <button class="btn" onclick="window.location.href='/chess\_staffview'">Chess(staff) Participants</button>  
  
 <button class="btn" onclick="window.location.href='/discuss\_stuview'">Discuss Throw(stu) Participants</button>  
 <button class="btn" onclick="window.location.href='/discuss\_staffview'">Discuss Throw(staff) Participants</button>  
  
 <button class="btn" onclick="window.location.href='/shotput\_stuview'">Shot Put(stu) Participants</button>  
 <button class="btn" onclick="window.location.href='/shotput\_staffview'">Shot Put(staff) Participants</button>  
  
 <button class="btn" onclick="window.location.href='/running\_stuview'">Running(100m)(stu) Participants</button>  
 <button class="btn" onclick="window.location.href='/running\_staffview'">Running(100m)(staff) Participants</button>  
  
 <button class="btn" onclick="window.location.href='/throw\_stuview'">Throw Ball(stu) Participants</button>  
 <button class="btn" onclick="window.location.href='/throw\_staffview'">Throw Ball(staff) Participants</button>  
  
 <button class="btn" onclick="window.location.href='/foot\_stuview'">Foot Ball(stu) Participants</button>  
 <button class="btn" onclick="window.location.href='/foot\_staffview'">Foot Ball(staff) Participants</button>  
  
 <button class="btn" onclick="window.location.href='/basket\_stuview'">Basket Throw(stu) Participants</button>  
 <button class="btn" onclick="window.location.href='/basket\_staffview'">Basket Throw(staff) Participants</button>  
  
 <button class="btn" onclick="window.location.href='/cricket\_stuview'">Cricket(stu) Participants</button>  
 <button class="btn" onclick="window.location.href='/cricket\_staffview'">Cricket(staff) Participants</button>  
  
 <button class="btn" onclick="window.location.href='/volley\_stuview'">Volley Ball(stu) Participants</button>  
 <button class="btn" onclick="window.location.href='/volley\_staffview'">Volley Ball(staff) Participants</button>  
  
 </div>  
 <button class="btn center-btn" onclick="window.location.href='/certificate'">Generate Certificate</button>  
 </div>  
</body> </html>

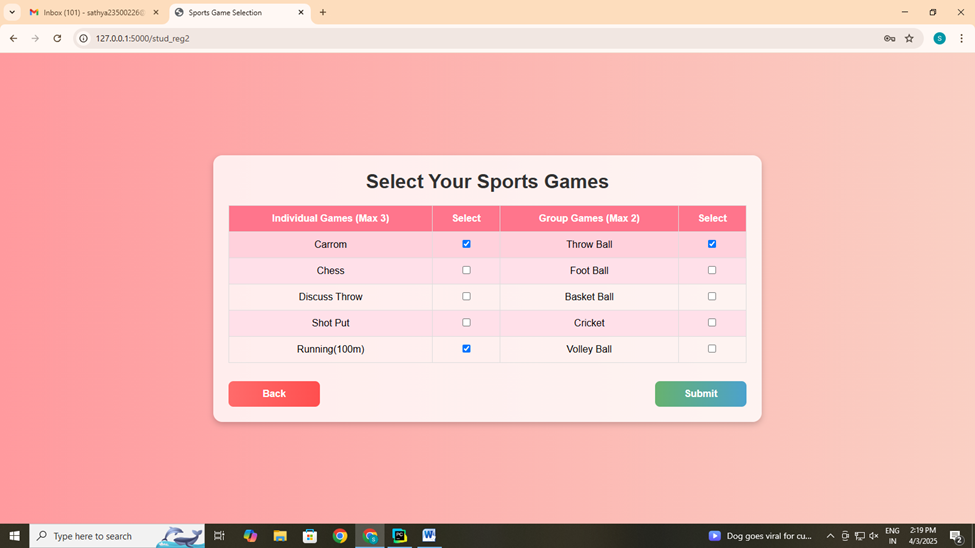
**SNAPSHOTS**

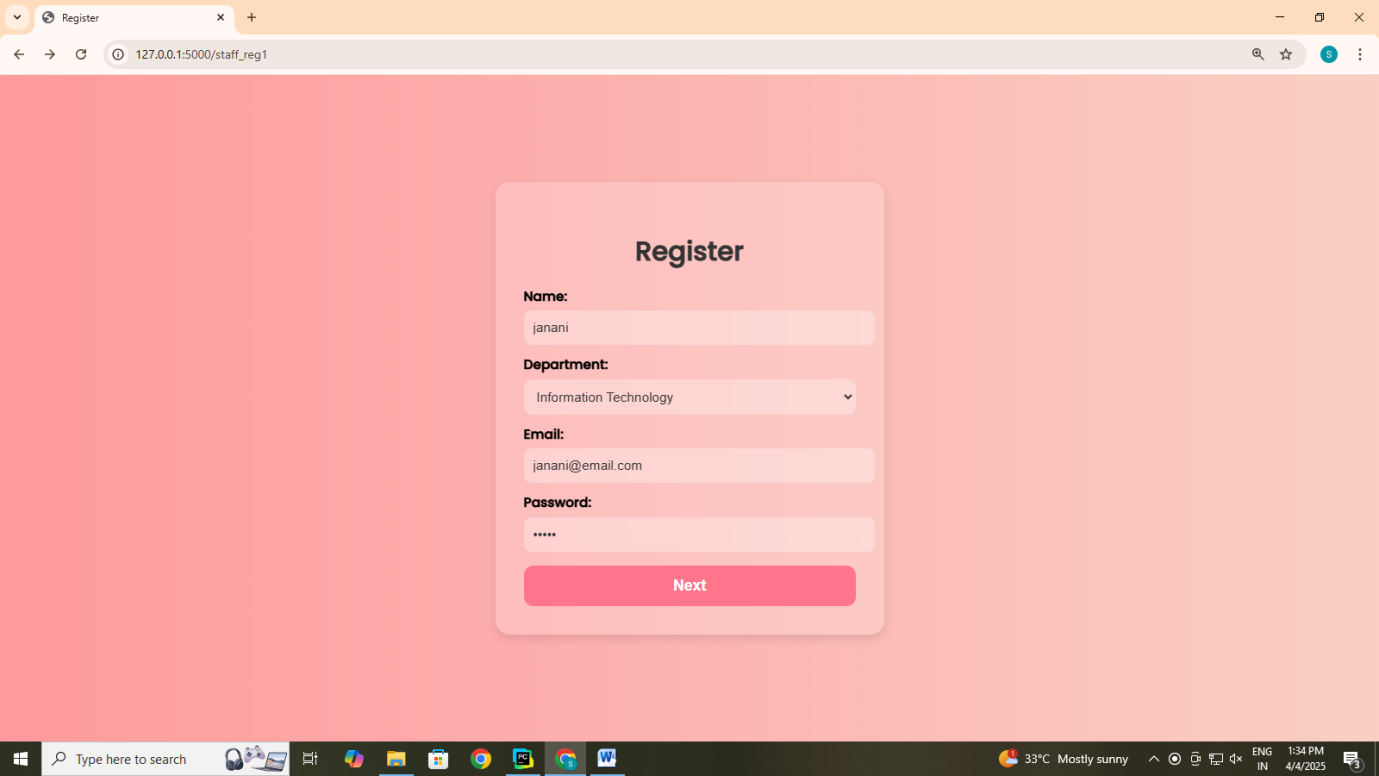


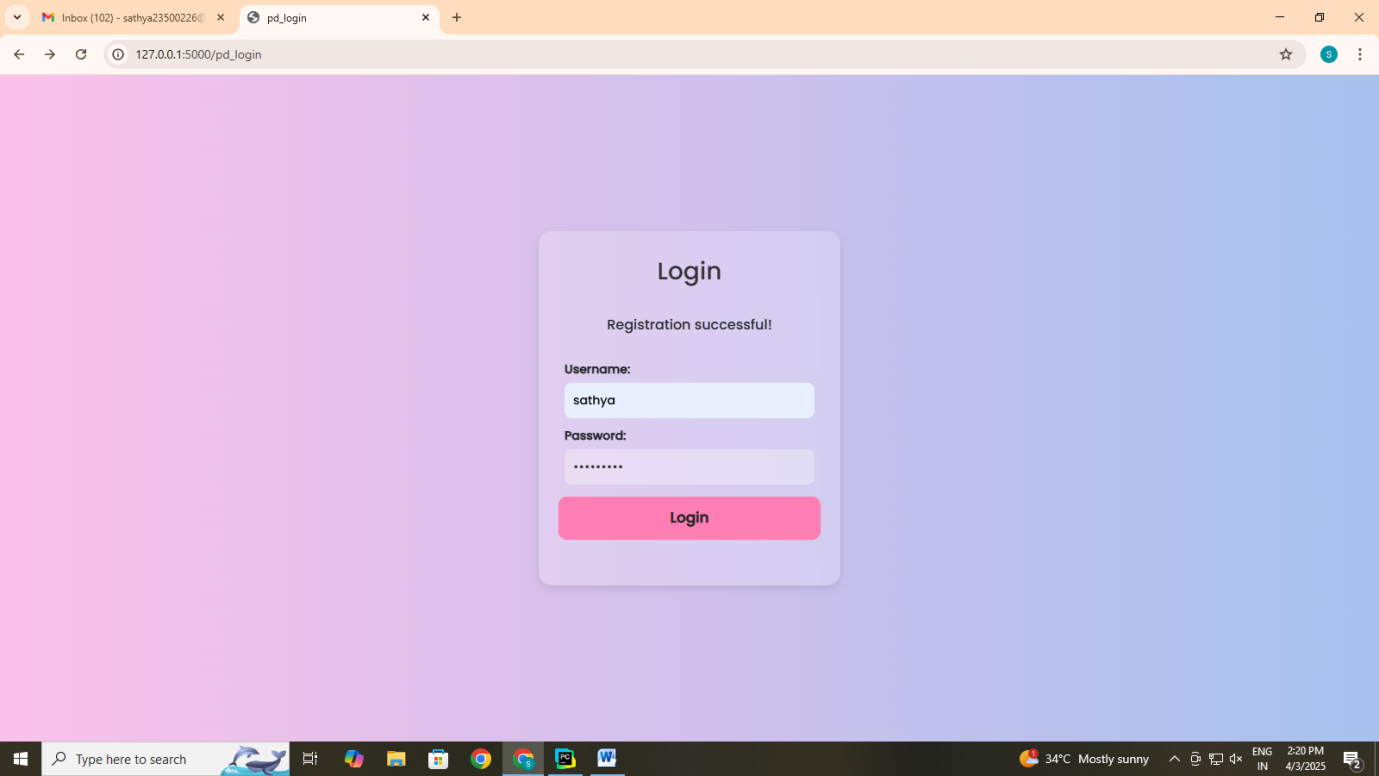
**Snapshots**

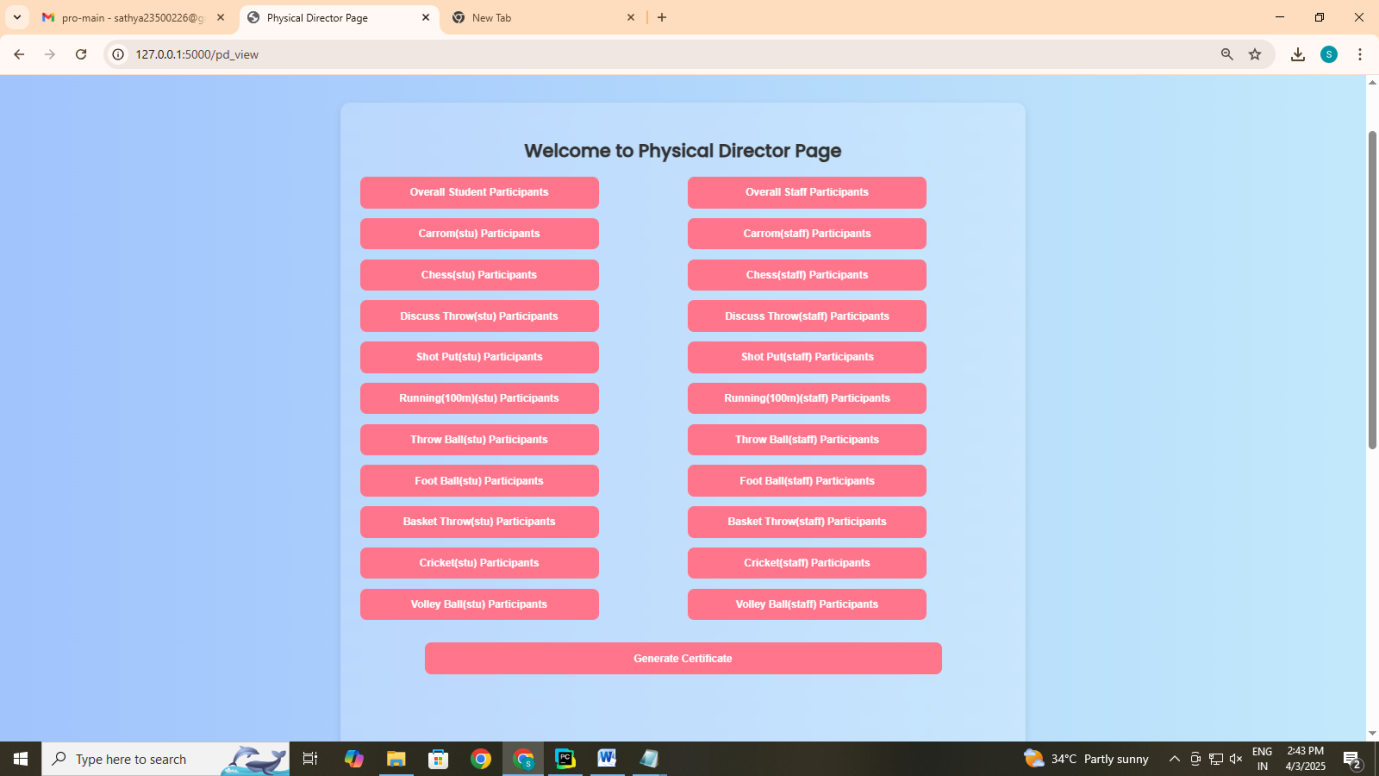


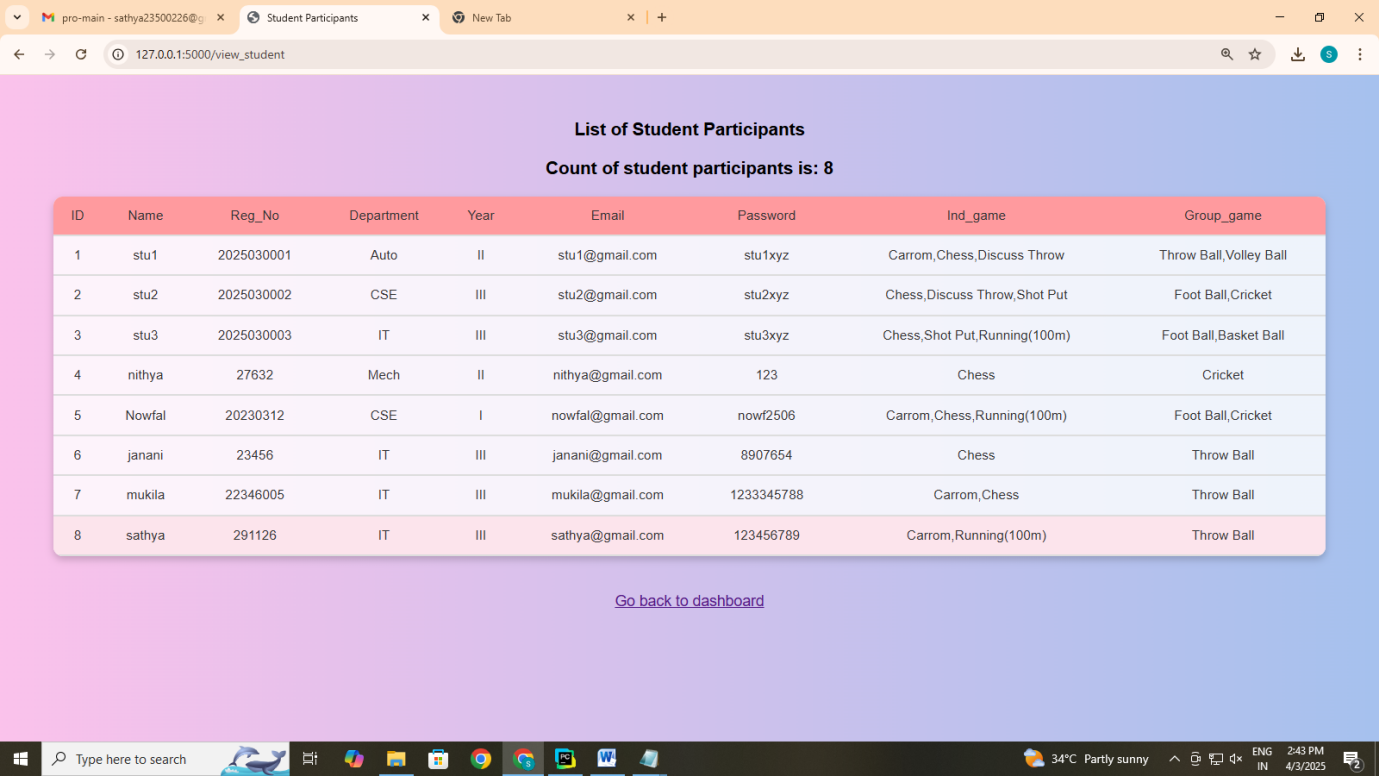
****

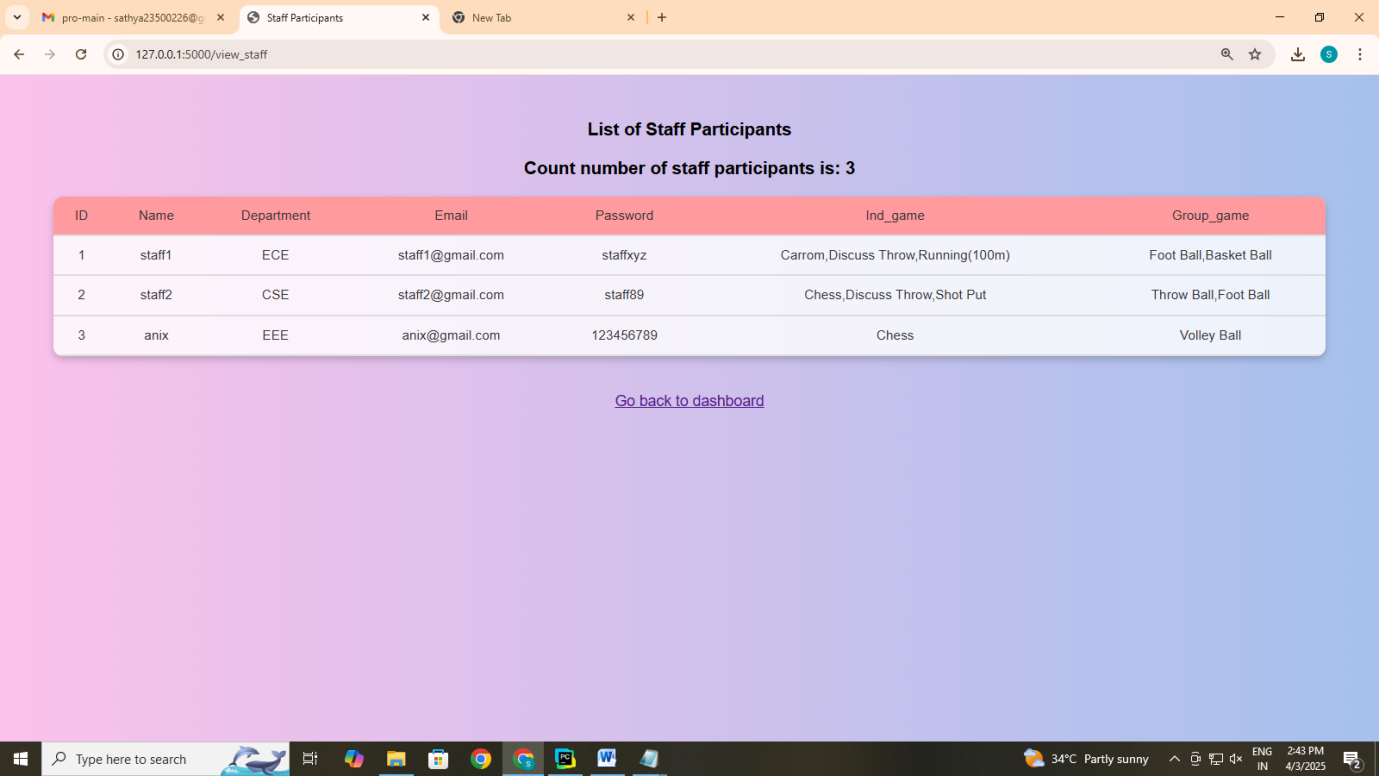
****

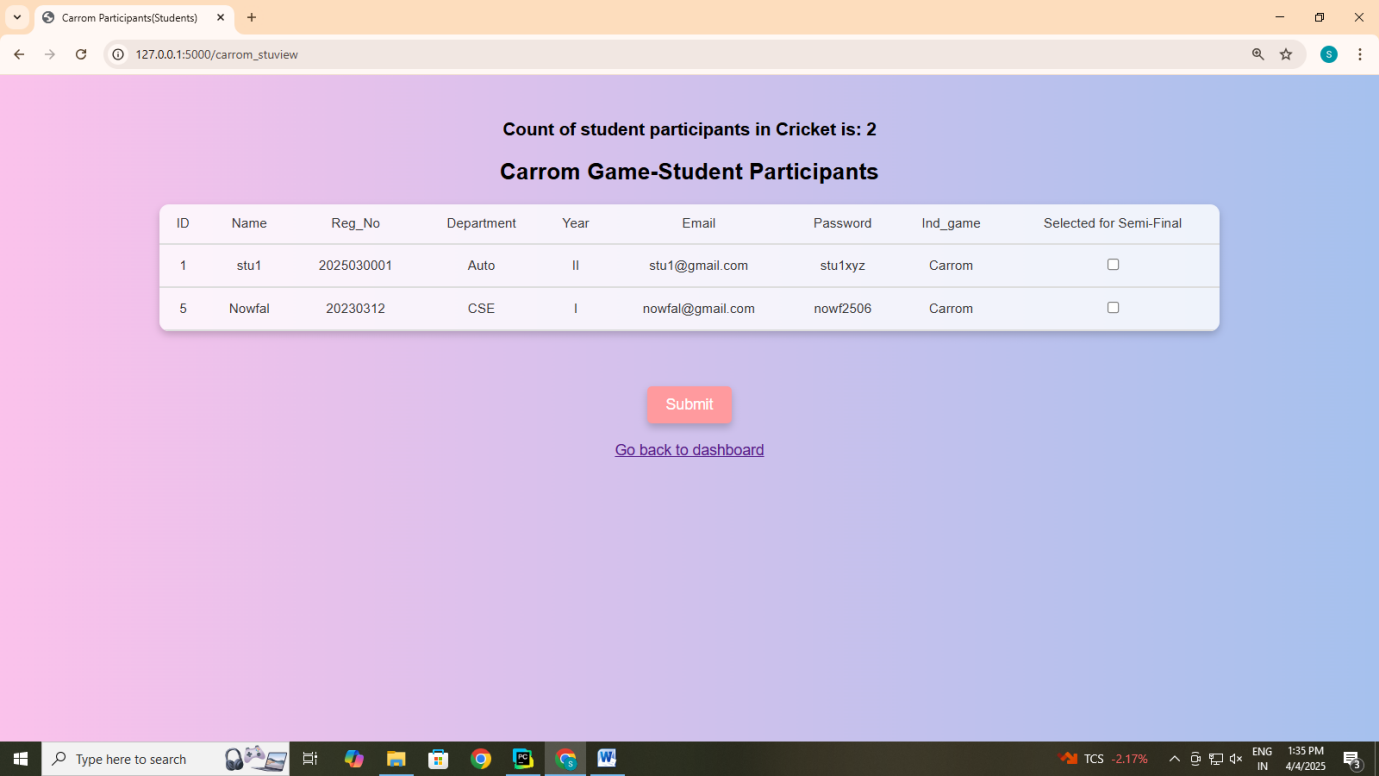


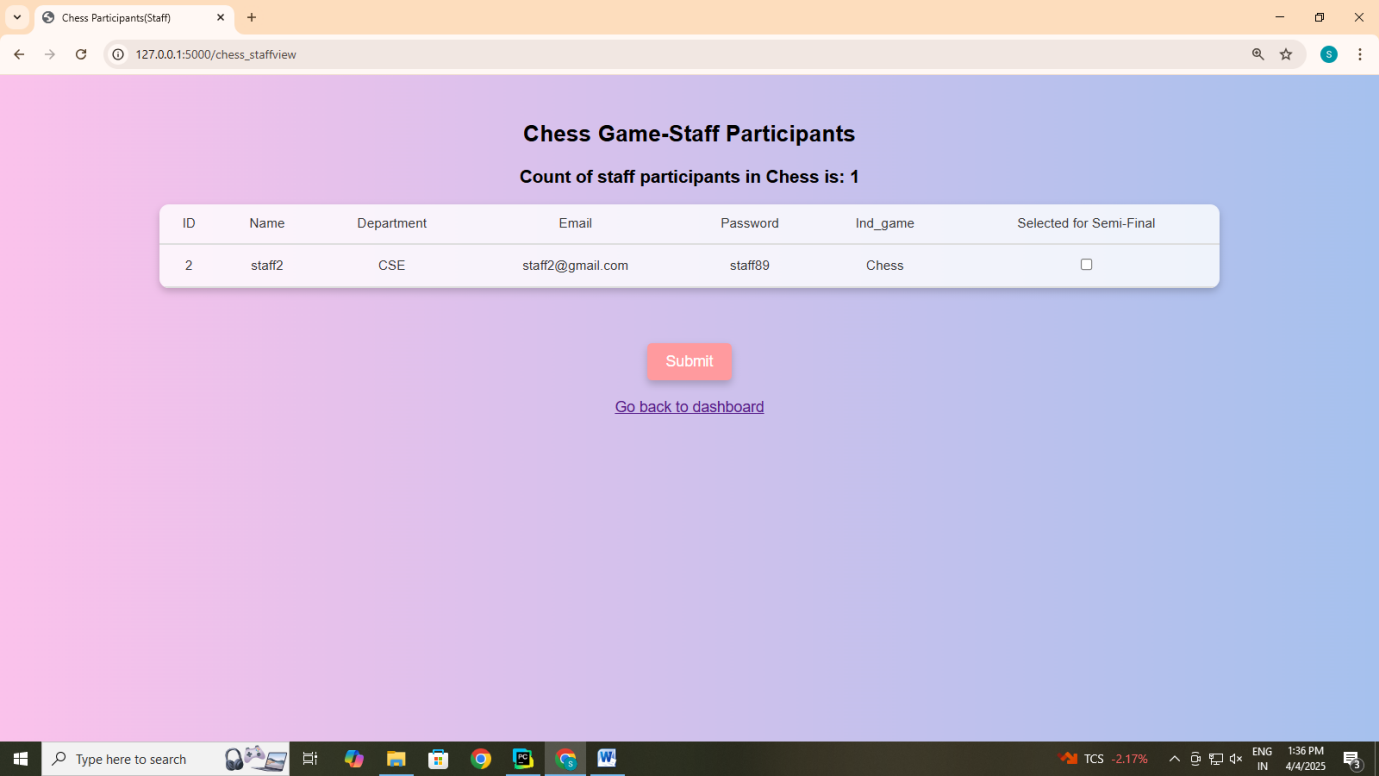


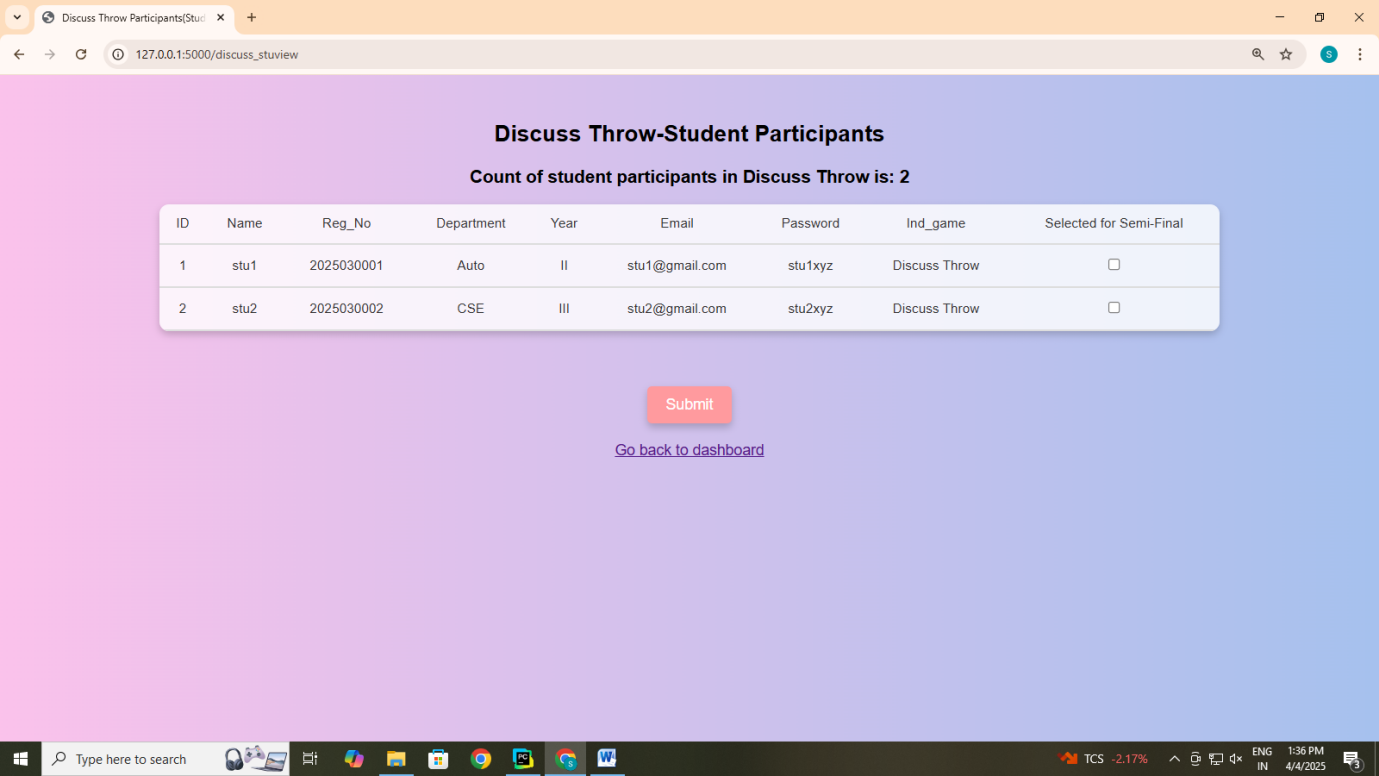


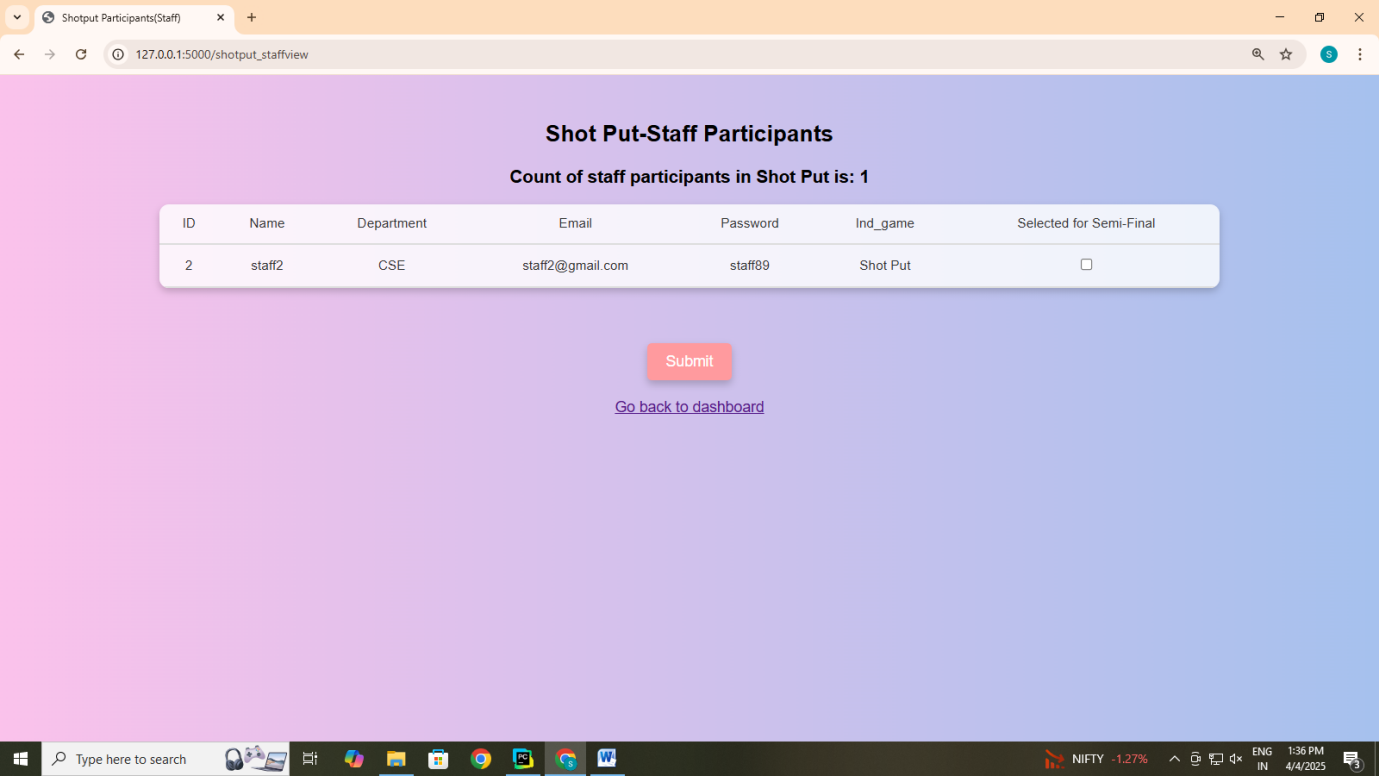


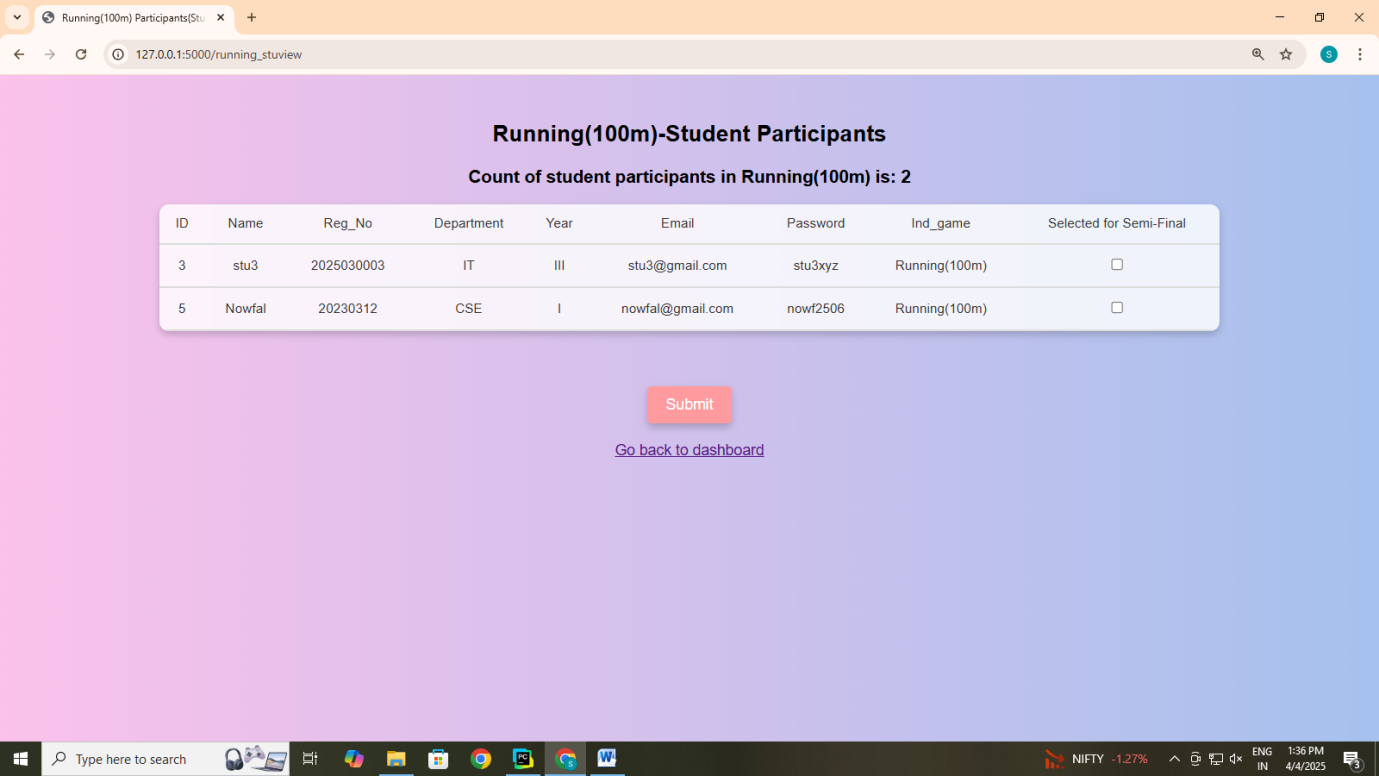


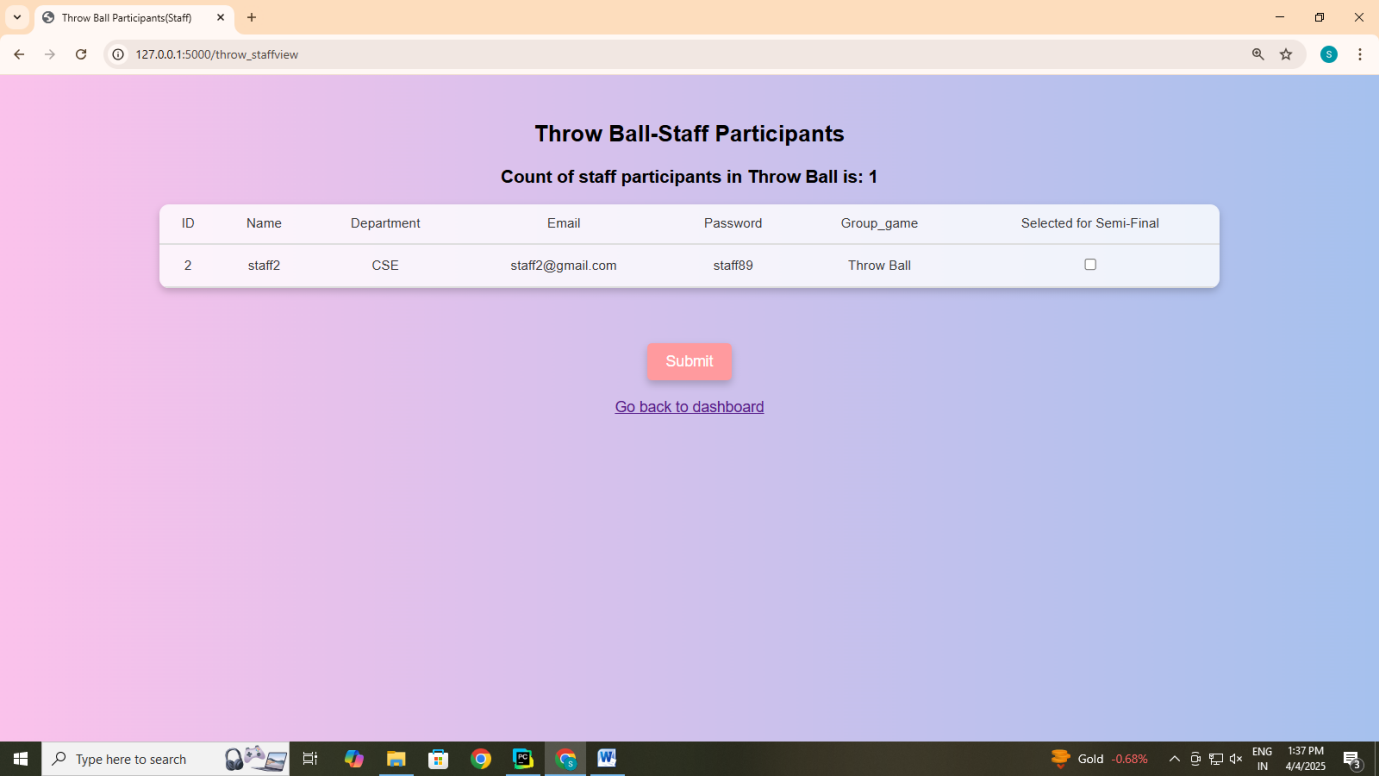


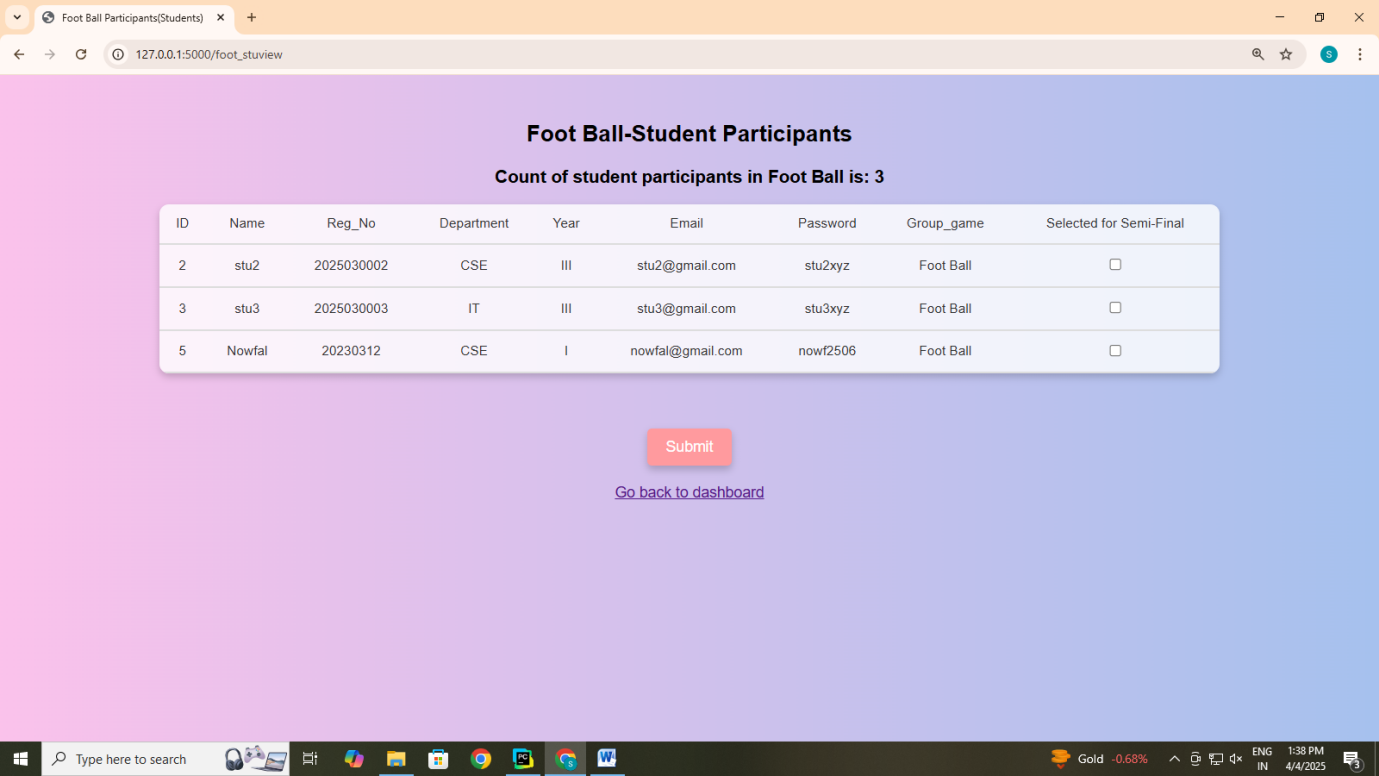














**CONCLUSION**

The Sports Day Management System efficiently organizes events, tracks participants, and updates results in real time. It enhances user experience with role-based access,and leaderboard tracking. Built using Flask, SQLite3, and a responsive UI, it reduces manual effort and ensures transparency. Participants can register, track their performance, and stay updated with event schedules. Future enhancements could include mobile support, real-time scoring, and AI-powered analytics for an even better experience.



**REFERENCE**

* <https://www.geeksforgeeks.org>
* <https://www.w3schools.com>
* <https://developer.mozilla.org>
* <https://www.codecademy.com>