INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR DEPARTMENT OF COMPUTER SCICENCE AND ENGINEERING

Database Management System Laboratory

Mini-Project Report

University Festival Management System Software

Advisor: Prof. Pabitra Mitra, Prof. KS Rao

IIT KHARAGPUR, AUGUST 2024

Member List

Name	Roll No
Ajay Kumar Dhakar	21CS30002
Ishan Ray	21CS10033
Swadhin Satyaprakash Majhi	21CS10067
Venkatesh Naresh	21CS10076
Jaykumar D Priyadarshi	21CS30026

Contents

1 University Festival Management System

The University Festival Management System is a comprehensive software solution revolutionizing the orchestration of university festivals. This centralized platform seamlessly integrates event planning, participant management, and financial oversight, offering a holistic approach to festival organization. Organizers can efficiently schedule diverse events, avoiding conflicts through a user-friendly interface allowing venue allocation and real-time updates.

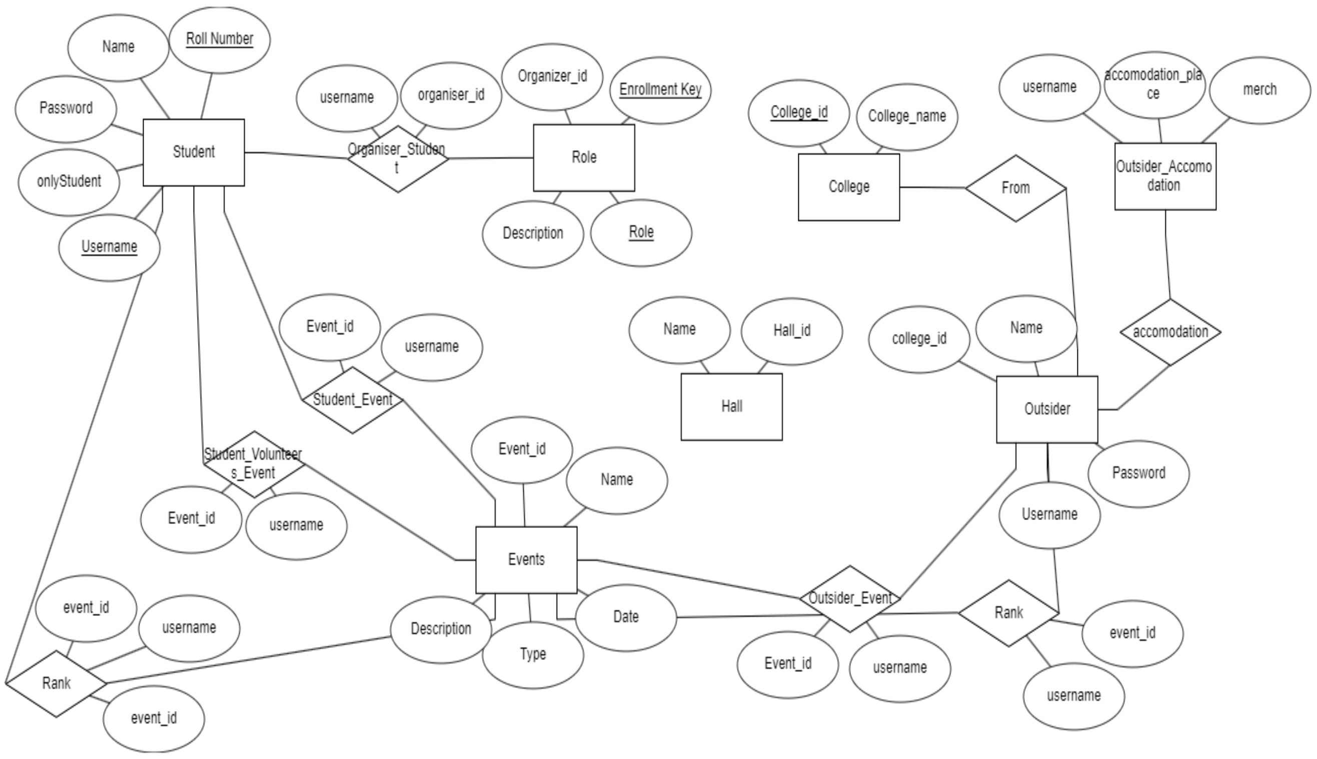
The system simplifies participant engagement by providing an intuitive online registration platform. It ensures effective budget management, enabling organizers to allocate funds for various festival expenses and track financial transactions. Integrated marketing tools enhance promotional activities, while the system facilitates ticketing and promotions for a wider audience reach.

Volunteer coordination becomes seamless as the system assists recruitment, role assignment, and hour tracking. Robust communication tools unify all stakeholders, disseminating crucial information and updates. Resource allocation is optimized, preventing bottlenecks and ensuring efficient equipment, venues, and personnel use.

The system generates comprehensive performance analytics, offering insights into attendance, participant feedback, and financial performance. With stringent security measures and scalability for festivals of varying sizes, the University Festival Management System stands as a customizable, transparent, and efficient solution for the successful execution of university festivals.

2 Database Design

The Entity-Relationship Diagram of our University Festival Management System is as follows.



The schemas are namely:

- Student Table

Student	
name	VARCHAR(255)
roll_number	CHAR(9)
username	VARCHAR(255)
password	VARCHAR(255)
onlyStudent	BOOLEAN

- This table facilitates efficient student data management. It stores names, ensures precise identification, and employs a unique roll_number as a distinctive identifier, maintaining data integrity. The username column, designated as the PRIMARY KEY, ensures unique login credentials for efficient data retrieval. While the password column secures authentication, it's advisable to implement encryption techniques for enhanced security. The only student column, a BOOLEAN flag, differentiates users with only the student role (true) from those with additional roles (false), enabling role-specific access. In essence, the table optimizes student information storage with a focus on data integrity, security, and role-based functionalities.
- College Table

College	
college_id	INT
college_name	VARCHAR(255)

- This table features a primary key college_id of integer type to uniquely identify colleges. The college_name column, a variable character string of up to 255 characters, stores the names of the colleges. This table efficiently manages basic information about colleges within a concise database structure.
- Outsider Table

College	
name	VARCHAR(255)
username	VARCHAR(255)
password	VARCHAR(255)
college_id	INT

This table captures outsider information with columns for name, username (set as PRIMARY KEY), and password. The inclusion of the college_id column establishes a FOREIGN KEY relationship with the College table, linking outsiders to specific colleges. This structured approach ensures data integrity and facilitates efficient management of outsider data within a concise database structure.

- Event Table

Event	
event_id	INT
name	VARCHAR(255)
date	DATE
description	VARCHAR(255)
type	VARCHAR(255)

The Event table represents events. It has columns for the event ID (which is the primary key), event name, date, type, and description.

- Organizer_Role Table

Organizer_Role	
organizer_id	INT
enrollment_key	INT
role	VARCHAR(255)
description	VARCHAR(255)

- The Organizer_Role table represents roles of organizers. It has columns for the organizer ID (which is the primary key), enrollment key (which is unique), role (which is also unique), and description.
- Organizer_to_Student table

Organizer_to_Student	
organizer_id	INT
username	VARCHAR(255)

The Organizer_to_Student table represents the relationship between organizers and students. It has columns for the organizer ID and username. Both columns are foreign keys, with the organizer ID referencing the Organizer_Role table and the username referencing the Student table.

winners_student

event_id	INT
username	VARCHAR(255)
rank	INT
FOREIGN KEY (event_id)	A4_Event(event_id)
FOREIGN KEY	A4_Student(username)

(username)	A4_Student(username)

winners_outsider

event_id	INT
username	VARCHAR(255)
rank	INT
FOREIGN KEY (event_id)	Event(event_id)
FOREIGN KEY (username)	Outsider(username)

Student_Participate_Event table

Student_Participate_Event	
event_id Username	INT VARCHAR(255)

- The Student_Participate_Event table represents the participation of students in events. It has columns for the event ID and username. Both columns are foreign keys, with the event ID referencing the Event table and the username referencing the Student table.
- Outsider_Participate_Event table

Outsider_Participate_Event	
event_id username	INT VARCHAR(255)

The Outsider_Participate_Event table represents the participation of outsiders in events. It has columns for the event ID and username. Both columns are foreign keys, with the event ID referencing the Event table and the username referencing the Outsider table.

- Student_Volunteer_Event table

Student_Volunteer_Event	
event_id username	INT VARCHAR(255)

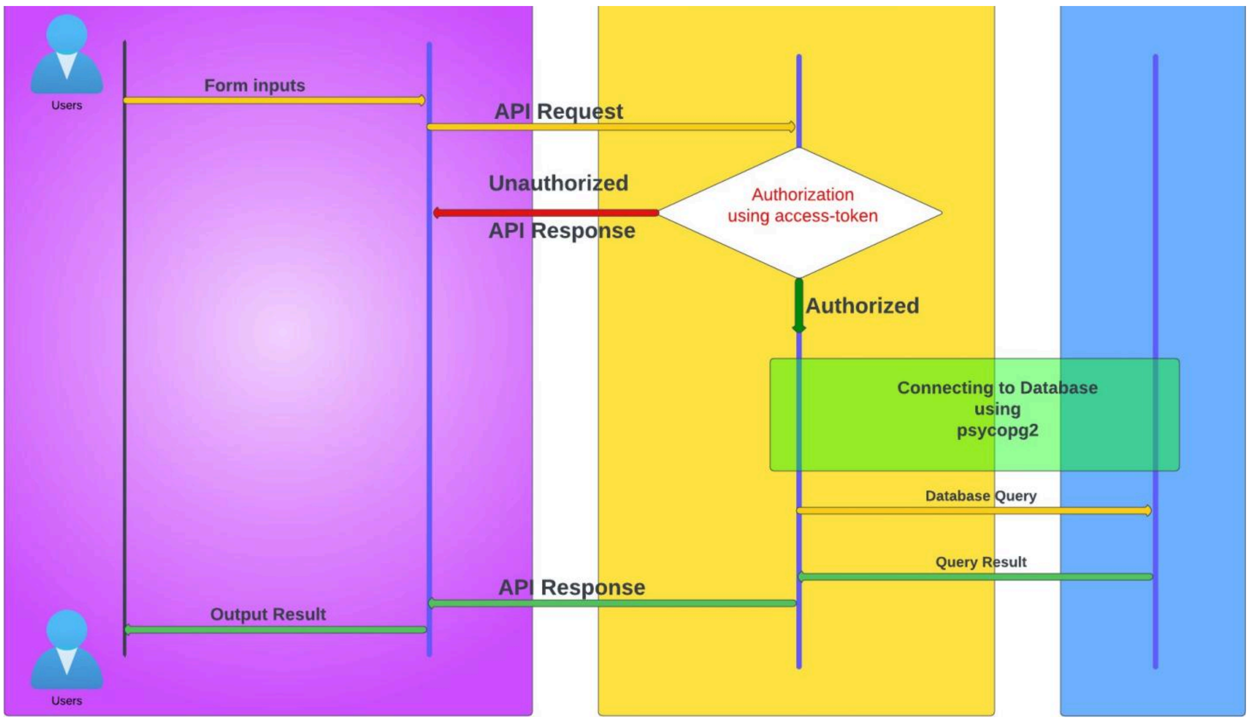
The Student_Volunteer_Event table represents the volunteering of students in events. It has columns for the event ID and username. Both columns are foreign keys, with the event ID referencing the Event table and the username referencing the Student table

3 Languages and Tools

In this section we will go through the languages, tools, and methodology we used to build this Fest Management System

This system consists of three segments, namely Frontend, Backend, and Database.

Frontend	HTML CSS
Backend	FastAPI
Database	Postgres



4 Implemented Workflows and Triggers

The workflow diagram of our system is as follows:

In the function delete_a_student

```
    # WE NEED TO DELETE THE STUDENT INSTANCE FROM ALL TABLES
    # query = f"DELETE FROM A4_Student_Winner_Event WHERE username = '{username}';"
    # cursor.execute(query)
    query = f"DELETE FROM A4_Student_Participate_Event WHERE username = '{username}';"
    cursor.execute(query)
    query = f"DELETE FROM A4_Student_Volunteer_Event WHERE username = '{username}';"
    cursor.execute(query)
    query = f"DELETE FROM A4_Student WHERE username = '{username}';"
```

In the function delete_a_organiser

```
    # WE NEED TO DELETE THE STUDENT INSTANCE FROM ALL TABLES
    query = f"DELETE FROM A4_Organizer_to_Student WHERE username = '{username}';"
    cursor.execute(query)
    query = f"DELETE FROM A4_Student WHERE username = '{username}';"
```

In the function delete_a_outsider(username):

```
    # WE NEED TO DELETE THE STUDENT INSTANCE FROM ALL TABLES
    # query = f"DELETE FROM A4_Outsider_Winner_Event WHERE username = '{username}';"
    # cursor.execute(query)
    # query = f"DELETE FROM A4_Outsider_Accomodation WHERE username = '{username}';"
    # cursor.execute(query)
    query = f"DELETE FROM A4_Outsider_Participate_Event WHERE username = '{username}';"
    cursor.execute(query)
    query = f"DELETE FROM A4_Outsider WHERE username = '{username}';"
```

5 Code Listings of Backend APIs

In this section we have listed all the APIs which we are using to implement the functionality of the system and also how the APIs query the database for the required data.

# Query 1: Register a student in A4_Student table

def register_student(name, roll_number, username, password, only_student):

# Query 2: Register an Outsider in A4_Outsider

def register_outsider(name, username, password, collegename):

# Query 3: Register an Organizer in both A4_Student and A4_Organizer_to_Student tables

def register_organizer(name, roll_number, username, password, only_student, enrollment_key):

# Query 4: Register an outsider for an event in A4_Outsider_Participate_Event

def register_outsider_for_event(event_id, username):

# Query 5: Fetch event name, type, and description from A4_Event table

def fetch_event_details():

# Query 6: Register a student for an event in A4_Student_Participate_Event

def register_student_for_event(event_id, username):

# Query 7: Allow student to volunteer in a given event in A4_Student_Volunteer_Event

def allow_student_to_volunteer(event_id, username):

```
# Query 8: Allow organizer to view all participants for a particular event

# A list of names and usernames are returned

def view_participants_for_event(desired_event_id):



# Query 9: Allow organizer to view all volunteers for an event

def view_volunteers_for_event(desired_event_id):



# Query 10: Allow organizer to change name, type, and description of an event

def change_event_details(desired_event_id, new_event_name, new_event_type, new_event_description):



# Query 11: Check if username exists in the A4_Student table

def check_username_in_student_table(username):



# Query 12: Check if username exists in the A4_Outsider table

def check_username_in_outsider_table(username):



# Query 13: Check if username exists in the A4_Organizer_to_Student table

def check_username_in_organizer_to_student_table(username):



# Query 14

def checkfororganiser(user=Organiser_l):

  # check for user.username

  # return 1 for username already exists

  # return 2 for roll number already exists

  # return 3 for enrollment key does not exist

  # return 0 for success
```

#Query 15

```
def checkforstudent(user=Student_l):
    # check for user.username
    # return 1 for username already exists
    # return 2 for roll number already exists
    # return 0 for success
```

#Query 16

```
def checkforoutsider(user=Outsider_l):
    # check for user.username
    # return 1 for username already exists
    # return 0 for success
```

#Query 17

```
def checkforuser(user=User_l):
    # check for user.username and user.password
    # return 1 for invalid username or password
    # return 0 for success
```

#Query 18:get all events

```
def getalleventstable():
```

#Query 19:get all college names

```
def getallcollegenames():
```

#Query 20: get participants of a event

def getalleventparticipants(event_id):

#Query 21: get details of a event

def geteventdetails(event_id):

#Query 22: get winners of a event

def geteventwinners(event_id):

#Query 23: allocate hall to outsiders

def allocate_outsider_to_hall(username):

#Query 24: get name of hall

def gethallname(username):

#Query 24: get student details

def getstudentdetails(username):

#Query 25: get details of organizer

def getorganiserdetails(username):

#Query 26: get all students

def getallstudents():

#Query 27: get all organisers

def getallorganisers():

#Query 28: get all outsides

def getalloutsiders():

#Query 29: delete a student

def delete_a_student(username):

#Query 30: delete an organizers

def delete_a_organiser(username):

#Query 31: delete an outsider

def delete_a_outsider(username):

THANK YOU