DATABASE MANAGEMENT SYSTEM

COLLEGE CLUB MANAGEMENT SYSTEM

MINIPROJECT

SECTION: F

NAME (1): MK VISHWAAS

SRN (1): PES1UG22CS312

NAME (2): MOHUL YP

SRN (2): PES1UG22CS360

1. Description about the statement (Short abstract)

We aim to simplify the management of college clubs by providing a platform that makes it easy for students to create, join, and participate in club activities, while also empowering club leaders to effectively manage their organizations.

1. ER Diagram

A paper with text on it

Description automatically generated

1. Relational Schema

A notebook with writing on it

Description automatically generated

1. DDL Commands

We have use different types of ddl commands :

CREATE DATABASE:

CREATE DATABASE ClubManagement;

CREATE TABLE:

Ex: CREATE TABLE Users ( user\_id INT PRIMARY KEY AUTO\_INCREMENT, username VARCHAR(50) UNIQUE NOT NULL, password VARCHAR(255) NOT NULL, role ENUM('admin', 'faculty', 'member') NOT NULL );

FOREIGN KEY Constraints:

Ex: FOREIGN KEY (club\_id) REFERENCES Clubs(club\_id)

TRIGGER:

Ex: CREATE TRIGGER before\_sponsor\_insert

BEFORE INSERT ON Sponsors

FOR EACH ROW

BEGIN

IF NEW.amount < 100 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Sponsor amount must be at least Rs.100';

END IF;

END;

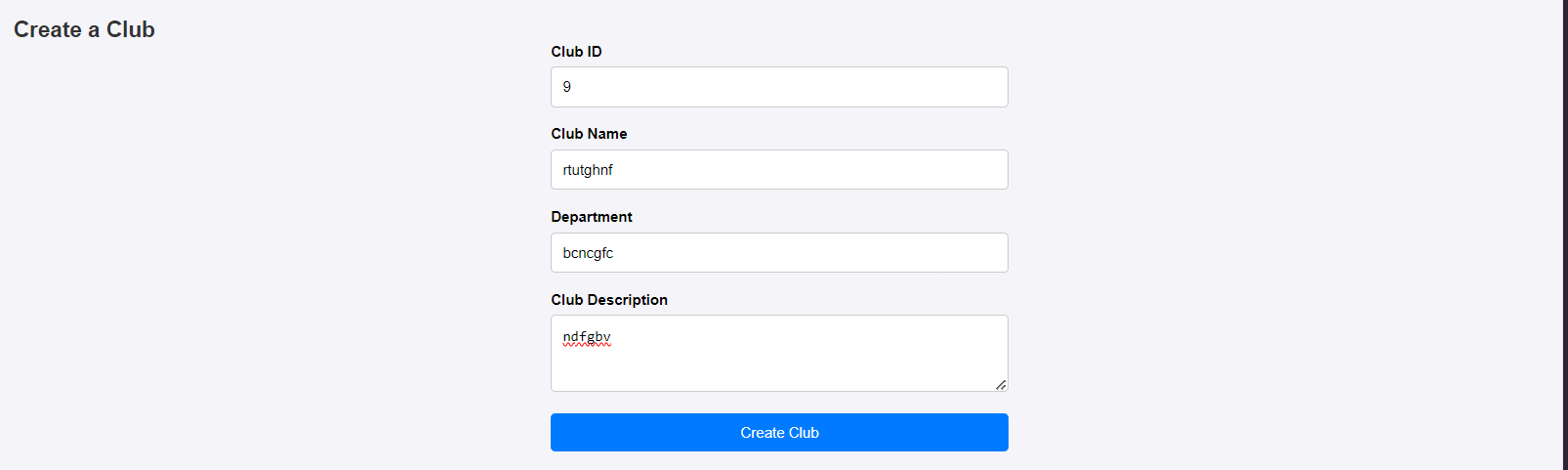
1. CRUD operation Screenshots

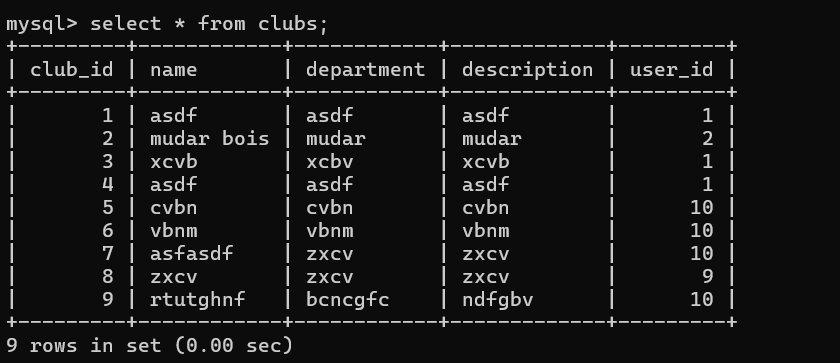
Create:

Before create:



After create:





SQL Query :  "INSERT INTO Clubs (club\_id, name, department, description, user\_id) VALUES (%s, %s, %s, %s, %s)",

                    (club\_id, name, department, description, user\_id)

Read:

A screenshot of a computer

Description automatically generated

SQL query : DELIMITER //

CREATE PROCEDURE GetEventsByClub(IN clubID INT)

BEGIN

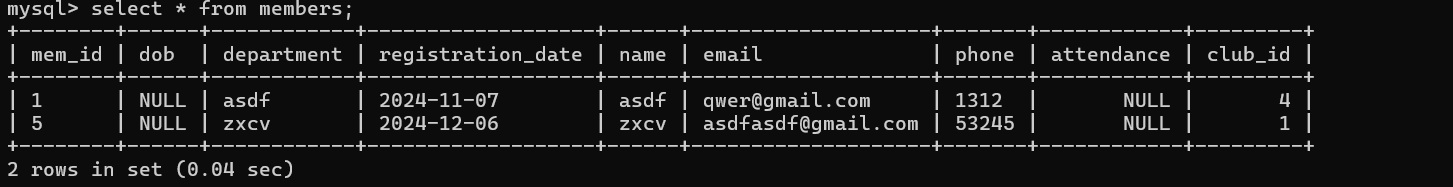
    SELECT \* FROM Events WHERE club\_id = clubID;

END //

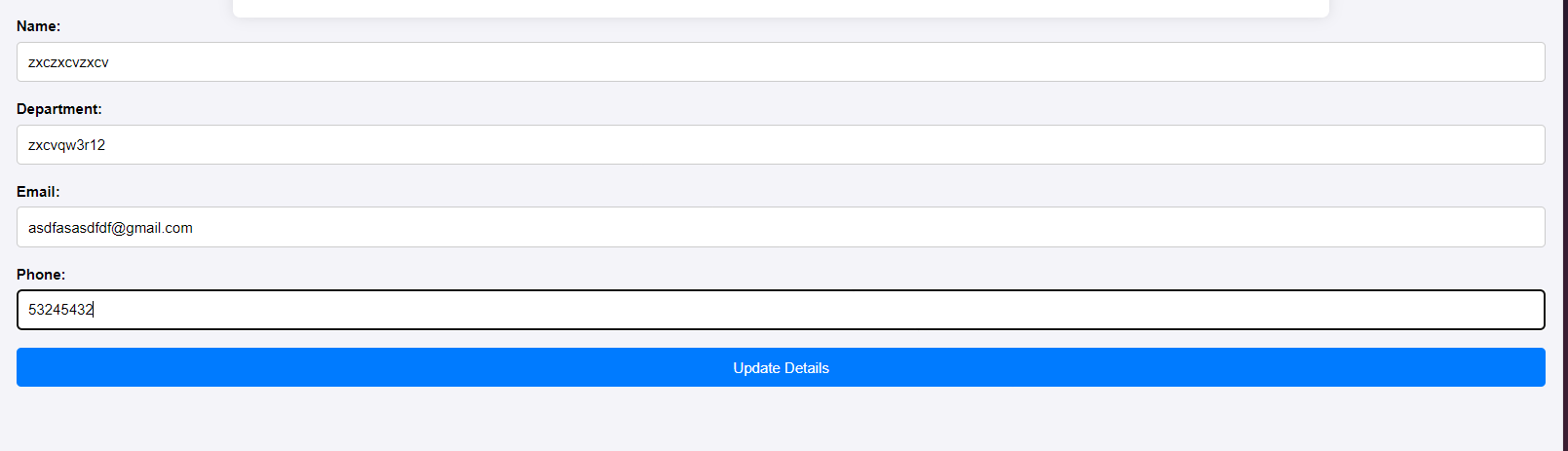
DELIMITER ;

Update:

Before update:



After update:





SQL Query :

UPDATE Members

SET

department = %s,

registration\_date = %s,

name = %s,

email = %s,

phone = %s,

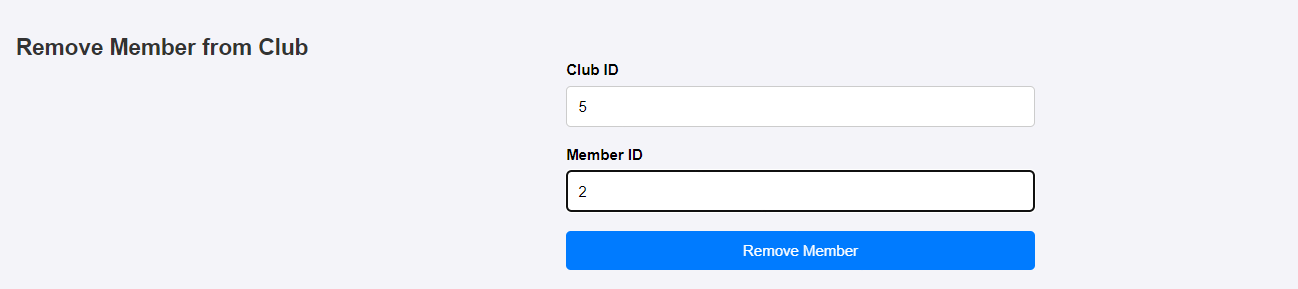
club\_id = %s

WHERE mem\_id = %s;

Delete:

Before delete :

  
After delete:





SQL query : DELETE FROM Members WHERE mem\_id = %s AND club\_id = %s", (member\_id, club\_id)

1. List of functionalities/features of the application and its associated screenshots using front end:

signup:

A screenshot of a computer

Description automatically generated

Login:

A computer screen shot of a login screen

Description automatically generated

Club admin:

A screenshot of a phone

Description automatically generated

Member module:

A screenshot of a computer

Description automatically generated

Faculty module:

A computer screen with a white and blue text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a login form

Description automatically generated

A screenshot of a login page

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Triggers, Procedures/Functions, Nested query, Join, Aggregate queries

Trigger to ensure sponsor amounts are at least Rs. 100.

Procedure to get all events for a specific club.

Nested query to get all member details of the club a faculty member belongs to.

Join and aggregate queries to find the total sponsorship amount for each club.

1. Code snippets for invoking the Procedures/Functions/Trigger

Procedure 1:

DELIMITER //

CREATE PROCEDURE GetEventsByClub(IN clubID INT)

BEGIN

SELECT \* FROM Events WHERE club\_id = clubID;

END //

DELIMITER ;

Procedure 2: DELIMITER //

CREATE PROCEDURE GetAllClubs()

BEGIN

SELECT \* FROM Clubs;

END //

DELIMITER ;

Trigger:

DELIMITER //

CREATE TRIGGER before\_sponsor\_insert

BEFORE INSERT ON Sponsors

FOR EACH ROW

BEGIN

IF NEW.amount < 100 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Sponsor amount must be at least Rs.100';

END IF;

END //

DELIMITER ;

Join and aggregate query:

SELECT Clubs.name AS club\_name, SUM(Sponsors.amount) AS total\_sponsorship

FROM Sponsors

JOIN Clubs ON Sponsors.club\_id = Clubs.club\_id

GROUP BY Clubs.name

ORDER BY total\_sponsorship DESC

Nested query:

SELECT name, email, department

FROM Members

WHERE club\_id = (

SELECT club\_id

FROM Faculty

WHERE faculty\_id = %s)

1. SQL queries(Create, Insert, Triggers, Procedures/Functions, Nested query, Join, Aggregate queries ) used in the project in the form of .sql file .

Available in the repo: [GitHub - galva2174/CCMS](https://github.com/galva2174/CCMS)

12: Github repo link:

[GitHub - galva2174/CCMS](https://github.com/galva2174/CCMS)