# ABSTRACT

The CMRU Clubs and Events Registration System is a web-based application designed to streamline and digitize the management of student engagement activities such as club memberships and event registrations at CMR University. Traditionally, these processes were handled manually through paper forms or spreadsheets, often leading to inefficiencies, data redundancy, and loss of important records. This project aims to address those issues by providing a centralized, secure, and user-friendly platform built using HTML, CSS, JavaScript, Python (Flask), and MySQL.

Students can create accounts, register for clubs, and enroll in university-hosted events, while their data is securely stored and organized in a structured relational database. The system automatically links students to their participations and memberships, ensuring seamless data flow and enabling real-time queries for analytics and reporting. Admins can retrieve information using SQL JOIN queries, and the system includes validation, duplicate checks, and role-based logic for consistent functionality.

The project also introduces secure login using password hashing and Flask sessions, and is designed to scale with future enhancements such as dashboards, email notifications, and mobile responsiveness. Overall, the system improves operational efficiency and enhances student participation visibility across campus activities, aligning well with digital transformation goals in higher education.

# SYSTEM DESIGN

The CMRU Clubs and Events Registration System has been designed with a focus on reliability, modularity, and scalability. The system architecture follows a layered model, comprising a frontend user interface, a Flask-based backend server, and a MySQL relational database. Each component is designed to interact with the others seamlessly, ensuring consistent data flow, robust validation, and secure user sessions.

### Frontend (User Interface)

The frontend is developed using standard web technologies — HTML, CSS, and JavaScript. It includes interactive web pages such as:

* Login and Registration forms
* Home and About sections
* Club enrollment page
* Event listing and registration page

Each page is designed for ease of use, with clear input fields and validation. CSS is used extensively to ensure consistency in layout, color schemes, responsiveness, and hover effects. JavaScript is responsible for dynamic form behavior, alert messages, and asynchronous communication with the backend using the fetch() API.

### Backend (Flask Server)

The backend uses Flask, a lightweight Python web framework. It handles:

* User authentication (login, register)
* Data processing (event/club registration)
* Session management
* Communication with the MySQL database

Routes such as /register, /login, /club\_register, and /event\_register are defined to handle POST requests from the frontend. Flask also uses Flask-Bcrypt to hash passwords for secure authentication and Flask-Session to manage user sessions. Responses from the backend are sent as JSON to be processed by the frontend.

### Database Design (MySQL)

The database is built using MySQL Workbench and consists of the following normalized tables:

* Users: Stores authentication credentials (user\_id, email, hashed password)
* Students: Contains detailed student profiles (student\_id, user\_id, name, department, semester, phone number)
* Clubs: Stores club information (club\_id, club\_name)
* Membership: Links students to clubs (student\_id, club\_id, role, join\_date)
* Events: Stores event details (event\_id, event\_name, date, description, club\_id)
* Participation: Tracks event enrollment (student\_id, event\_id, registration\_date)

Foreign keys are used to maintain referential integrity. For instance, Students.user\_id references Users.user\_id, while Membership and Participation use student\_id and club\_id or event\_id to link relationships.

### ER Diagram and Normalization

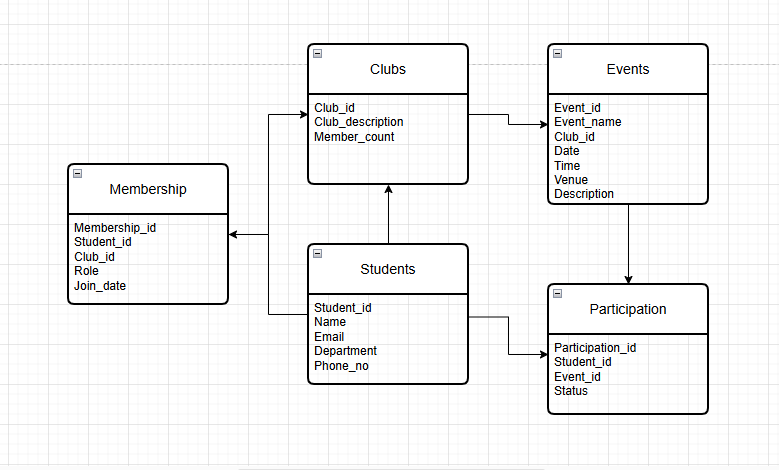
The database schema is based on an Entity-Relationship Diagram (ERD) that clearly defines the relationships between entities. It is normalized up to Third Normal Form (3NF) to eliminate redundancy and ensure efficient storage. For example:

* Users and Students are separated to keep authentication logic independent from academic data.
* Join tables (Membership and Participation) allow many-to-many relationships between students and clubs/events.

### Security and Validation

The system uses hashed passwords, form validations, and duplicate checks to maintain security and data integrity. Email uniqueness is enforced, and SQL queries are prepared with parameterized statements to prevent SQL injection.

# ER DIAGRAM



This ER (Entity-Relationship) diagram outlines a database structure for managing college clubs, students, memberships, and events. Here’s a breakdown:

### Entities and Relationships

1. Clubs: Stores club details (Club\_id, description, member count).
2. Students: Tracks student info (Student\_id, name, email, department).
3. Membership: Acts as a junction table linking students to clubs, recording their Role (e.g., member/leader) and Join\_date.
4. Events: Manages event details (Event\_id, name, date/time, venue) linked to clubs via Club\_id.
5. Participation: Another junction table connecting students to events, with Status (e.g., registered/attended).

Key Relationships

One - to- Many:

- A club (`Clubs`) can host multiple events (`Events`).

- A student (`Students`) can join multiple clubs (`Membership`) and participate in multiple events (`Participation`).

Many - to - Many

- Students ↔ Clubs (via `Membership`).

- Students ↔ Events (via `Participation`).

# DATABASE SCHEMA

The database schema for the CMRU Clubs and Events Registration System is designed to efficiently store, manage, and retrieve information related to students, their club memberships, and event participations. It follows a relational model using MySQL as the database engine and is normalized up to the Third Normal Form (3NF) to ensure minimal redundancy and maximum consistency.

### Core Design Philosophy

The primary goal of the database schema is to create a centralized and structured repository of all entities involved in the system. It supports CRUD (Create, Read, Update, Delete) operations for:

* User authentication
* Club and event management
* Student registrations
* Participation tracking

Each table is linked logically through foreign keys, ensuring referential integrity and data accuracy. The schema was also designed to scale with ease, allowing the addition of new clubs, events, or features with minimal changes.

### Tables and Their Relationships

#### Users Table

The Users table handles login credentials and basic authentication data. It includes:

* user\_id (Primary Key)
* username
* email (Unique)
* password\_hash (hashed using Bcrypt)

This table is linked to the Students table to keep authentication separate from academic and profile data.

#### Students Table

The Students table stores detailed student information:

* Student\_id (Primary Key)
* User\_id (Foreign Key referencing Users
* Name
* Email
* Phone\_no
* Department
* Semester

This ensures that every student has a profile that can be connected to multiple clubs and events without redundancy.

#### Clubs Table

The Clubs table maintains the list of all university-recognized clubs:

* Club\_id (Primary Key)
* club\_name (Unique)

This acts as a reference point for linking both memberships and events.

#### Membership Table

The Membership table captures the association between students and clubs:

* Membership\_id (Primary Key)
* Student\_id (Foreign Key referencing Students)
* Club\_id (Foreign Key referencing Clubs)
* Role (e.g., Member, Leader)
* Join\_date

This represents a many-to-many relationship between students and clubs and is essential for tracking involvement and hierarchy within each club.

#### Events Table

The Events table stores information about various activities conducted:

* Event\_id (Primary Key)
* event\_name (Unique)
* Description
* event\_date
* Club\_id (Foreign Key)

This helps associate each event with its organizing club.

#### Participation Table

The Participation table records students who have enrolled in events:

* Participation\_id (Primary Key)
* Student\_id (Foreign Key)
* Event\_id (Foreign Key)
* Registration\_date (Timestamp)

This table handles the many-to-many relationship between students and events, allowing multiple students to register for multiple events.

### Normalization and Integrity

All tables are normalized to 3NF:

* 1NF: Each column holds atomic values (no repeating groups).
* 2NF: Every non-key attribute is fully functionally dependent on the primary key.
* 3NF: No transitive dependencies exist.

Constraints like UNIQUE, NOT NULL, and FOREIGN KEY are used to maintain data integrity, avoid duplication, and ensure that related records are valid.

### Scalability and Flexibility

The schema can easily be extended to support features like:

* Admin roles and permissions
* Event attendance tracking
* Certificates and activity logs
* Feedback systems

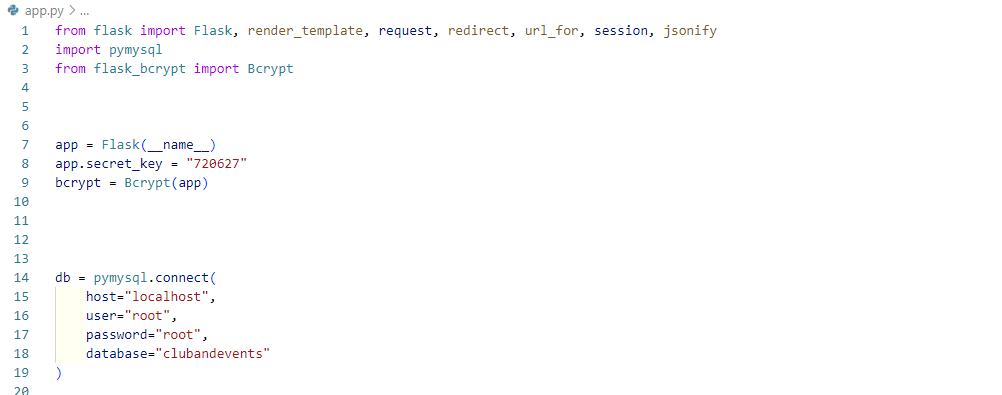
Its clear structure and logical flow make it suitable for both student-facing and admin-facing functionalities.

# DATABASE IMPLEMENTATION

The data implementation phase of the CMRU Clubs and Events Registration System played a crucial role in translating the database schema into a working, interactive, and fully connected application. This phase involved not only populating the database with meaningful, testable data but also ensuring that the integration between the frontend (forms), backend (Flask routes), and MySQL database functioned seamlessly.

### Establishing the Database Connection

The project uses MySQL as the relational database, and the connection was established using pymysql, a Python library that enables interaction between Flask and MySQL. The database connection string in the Flask app (app.py) was configured to include host, username, password, and database name.



Once the connection was established, a cursor object was created to execute SQL statements. All data manipulation tasks such as insertions, selections, and deletions were performed using this cursor.

### User Registration and Authentication

When a user registers using the register.html page, the entered data (full name, email, and password) is sent to the Flask backend using a POST request. In the Flask route:

* The password is hashed using Flask-Bcrypt to ensure security.
* The user’s details are then inserted into the Users table.
* After insertion, the newly generated user\_id is retrieved and used to populate the Students table.

If the student had already registered earlier through club or event forms, duplicate checks were in place to avoid redundancy.





### Club Registration Data Flow

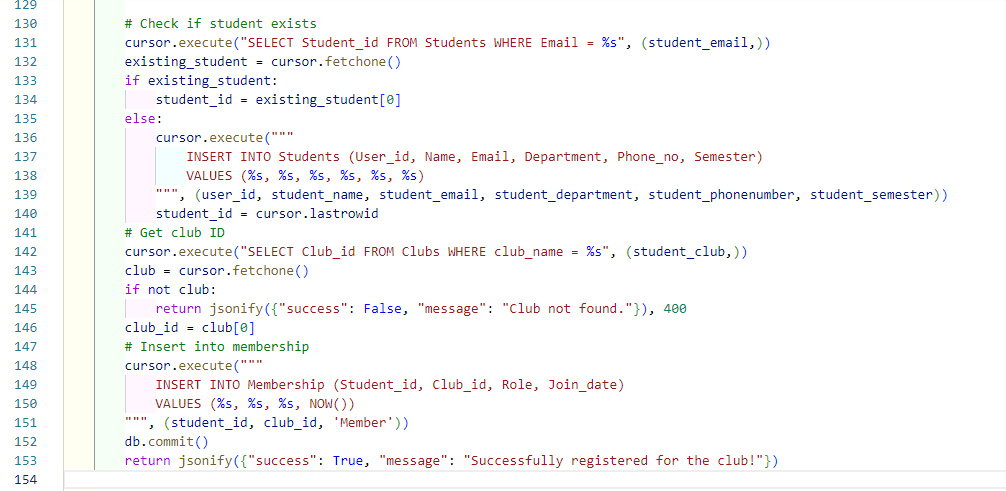
The club registration form captures additional student details such as:

* Department
* Semester
* Phone number
* Selected club

When submitted, this data flows to the Flask backend, where:

* The system first checks if the student exists in the Students table based on email.
* If not, a new record is inserted into Students, referencing the correct user\_id.
* The selected club’s club\_id is fetched.
* A new membership record is inserted into the Membership table.

This ensures the relationship between students and clubs is properly mapped and stored.



### 

### Event Registration and Participation Table

The event registration form works similarly. It collects student identity and enrollment details. Upon submission:

* The backend checks if the student exists in the Students table.
* If not, their details are inserted.
* The specific event’s event\_id is fetched based on the selected event.
* A new row is inserted into the Participation table, tracking that the student is enrolled for that event.



# TABLE STRUCTURE

The CMRU Clubs and Events Registration System is powered by a well-organized relational database, designed to manage information efficiently through multiple interconnected tables. The table structure was built on the principles of data normalization, integrity, and scalability, ensuring that the system could handle complex relationships among students, clubs, events, and their associated activities.

The database comprises six major tables: Users, Students, Clubs, Membership, Events, and Participation. Each table has a clearly defined primary key, relevant attributes, and foreign key constraints to preserve relational integrity.

### Users Table

The Users table is the first point of entry for any user into the system. It contains:

* user\_id (Primary Key): Uniquely identifies each user.
* username: Stores the name entered during registration.
* email: A unique identifier for login.
* password\_hash: Stores the securely hashed password.

This table is essential for authentication and session management, and acts as a parent to the Students table.

### Students Table

The Students table holds detailed profile information about each registered student. Attributes include:

* Student\_id (Primary Key)
* User\_id (Foreign Key referencing Users)
* Name
* Email
* Phone\_no
* Department
* Semester

This separation allows the system to manage academic and contact details independently of login credentials. The table ensures that each student is linked to a valid user in the Users table, maintaining data consistency.

Clubs Table

The Clubs table stores a list of all active clubs:

* Club\_id (Primary Key)
* Club\_name (Unique and descriptive)
* Club\_description
* Member\_count

This acts as a reference table and plays a crucial role in connecting with Events and Membership.

### Membership Table

The Membership table is a junction table representing the many-to-many relationship between students and clubs. It includes:

* Membership\_id (Primary Key)
* Student\_id (Foreign Key referencing Students)
* Club\_id (Foreign Key referencing Clubs)
* Role (e.g., Member, President)
* Join\_date

Each record in this table reflects a unique membership instance and the role the student holds in the club.

### Events Table

The Events table stores all event details hosted by various clubs. It contains:

* Event\_id (Primary Key)
* event\_name (Unique)
* Description
* Date
* Time
* Venue
* Club\_id (Foreign Key referencing Clubs)

This structure allows clubs to organize multiple events, and each event is linked back to the club responsible for it.

### Participation Table

The Participation table manages student enrollments in events:

* Participation\_id (Primary Key)
* Student\_id (Foreign Key referencing Students)
* Event\_id (Foreign Key referencing Events)

This is another many-to-many junction table and ensures that every event registration is uniquely identified and linked to the respective student and event.