EVENT MANAGEMENT SYSTEM

CONTENT

DBMS

Database Management System.

It is crucial for managing large volumes of data generated in various sectors, including event management. A robust DBMS ensures data integrity, security, and easy accessibility

What is DBMS

A Database Management System (DBMS) is a special kind of software that helps us store, organize, and manage data in a computer.

Instead of writing everything on paper or in random files, we can keep all the data in a structured and safe way using a DBMS. It allows us to add, view, change, or delete data easily whenever we want.

Abstract

The Event Management System (EMS) is a software that helps make planning and managing events easier and faster. It is built using Python for the user interface and SQL for storing data safely. Users can register participants, create event schedules, and send messages through the system–without needing to know programming.

The system saves time by automating tasks and reducing mistakes. It works well for both small and large events, and helps organizers and participants stay informed with clear schedules and communication tools. EMS improves overall event experience, saves effort, and helps events run smoothly with less manual work.

INTRODUCTION

Managing events like meetings, conferences, and public gatherings involves many tasks such as planning, registration, and scheduling. Doing all this manually takes a lot of time and effort.

The Event Management System (EMS) solves these problems by using Python and SQL to automate the work. Organizers can easily create events, manage schedules, track participants, and send important messages. The Python interface makes the system easy to use, and SQL keeps event data organized and safe.

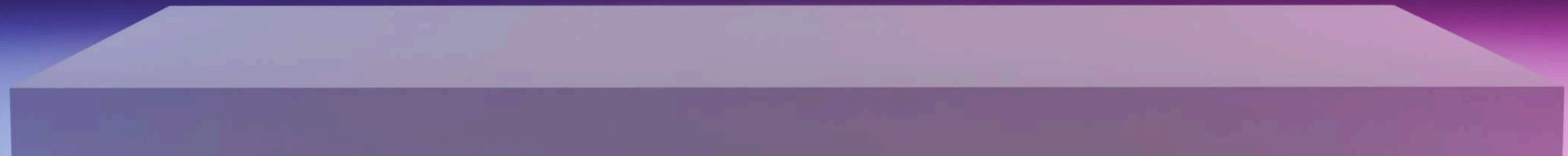
EMS is useful for both small and large events, helping reduce mistakes, save time, and keep participants updated. It makes event planning simple, professional, and more efficient.

Problem Statement

Event organization in colleges or institutions often involves:

- Manual registration using paper forms.
- Unclear schedules or miscommunication with participants.
- Repeated data entry errors and duplicate entries.
- Difficulty in tracking participants or sending updates.
- Time-consuming reporting and attendance tracking.

Hence, a digital solution is required that automates event handling and is easy to use for both admins and participants.



Database Design

1. Entities and Their Tables

1. Admin

Field Name	Data Type	Constraints
`admin_id`	INT	Primary Key, AUTO_INCREMENT
`username`	VARCHAR(50)	UNIQUE, NOT NULL
`password`	VARCHAR(100)	NOT NULL

2. Events

Field Name	Data Type	Constraints
`event_id`	INT	Primary Key, AUTO_INCREMENT
`event_name`	VARCHAR(100)	NOT NULL
`event_type`	VARCHAR(50)	NOT NULL
`location`	VARCHAR(100)	NOT NULL
`event_date`	DATE	NOT NULL
`fee`	DECIMAL(6,2)	DEFAULT 0
`capacity`	INT	NOT NULL
`description`	TEXT	

3. Participants

Field Name	Data Type	Constraints
`participant_id`	INT	Primary Key, AUTO_INCREMENT
`name`	VARCHAR(100)	NOT NULL
`mobile`	VARCHAR(10)	UNIQUE, NOT NULL
`email`	VARCHAR(100)	UNIQUE, NOT NULL
`branch`	VARCHAR(50)	

4. Registrations

Field Name	Data Type	Constraints
`registration_id`	INT	Primary Key, AUTO_INCREMENT
`participant_id`	INT	Foreign Key → `Participants(participant_id)`
`event_id`	INT	Foreign Key → `Events(event_id)`
`timestamp`	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

5. Branch Table

Column Name	Data Type	Description
`branch_id`	INT	Primary Key
`branch`	VARCHAR(50)	Branch name (e.g., CSE)

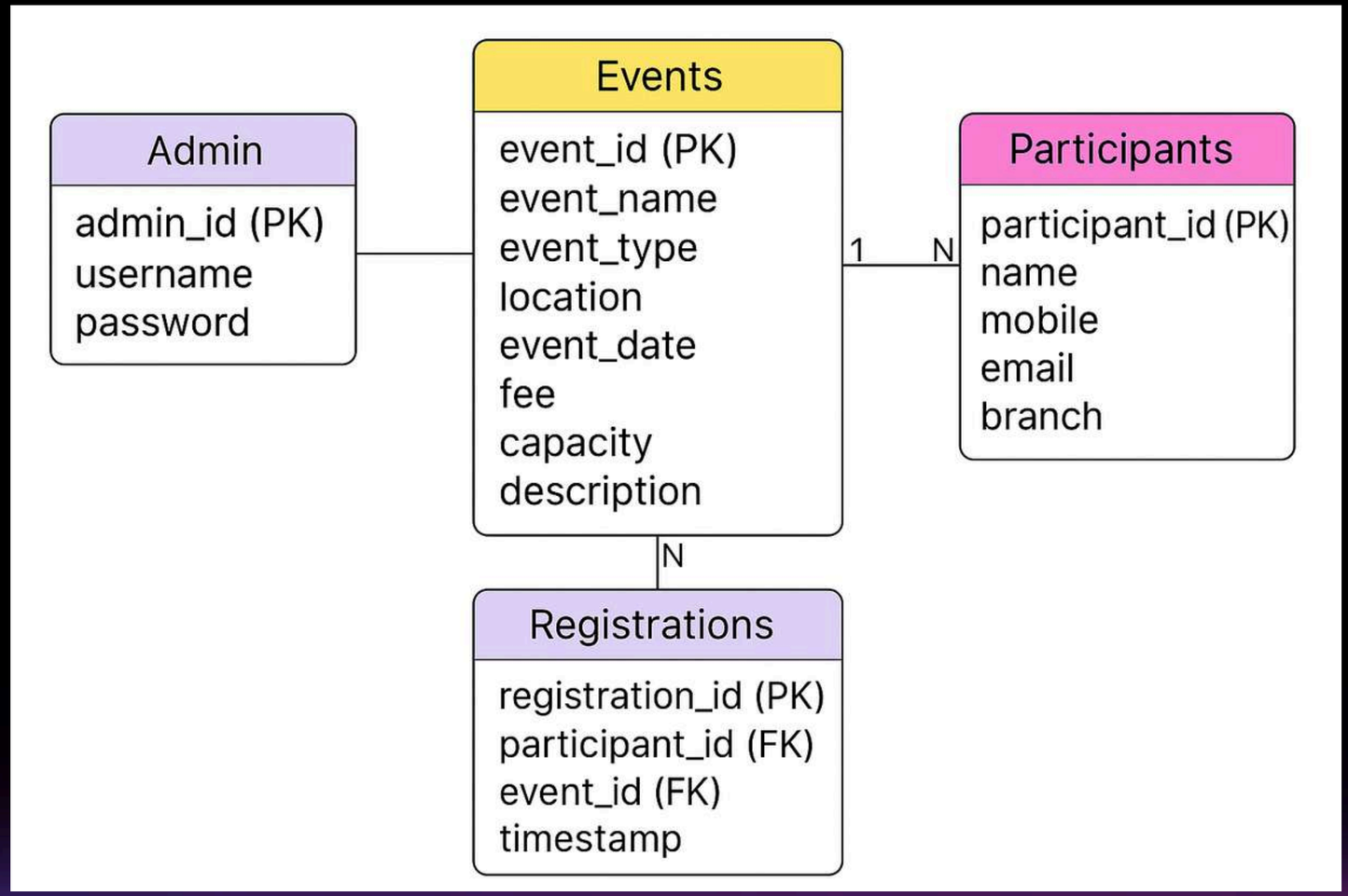
✅ 6. Event_Type Table

Column Name	Data Type	Description
`type_id`	INT	Primary Key
`event_type`	VARCHAR(50)	E.g., Seminar, Workshop, etc.

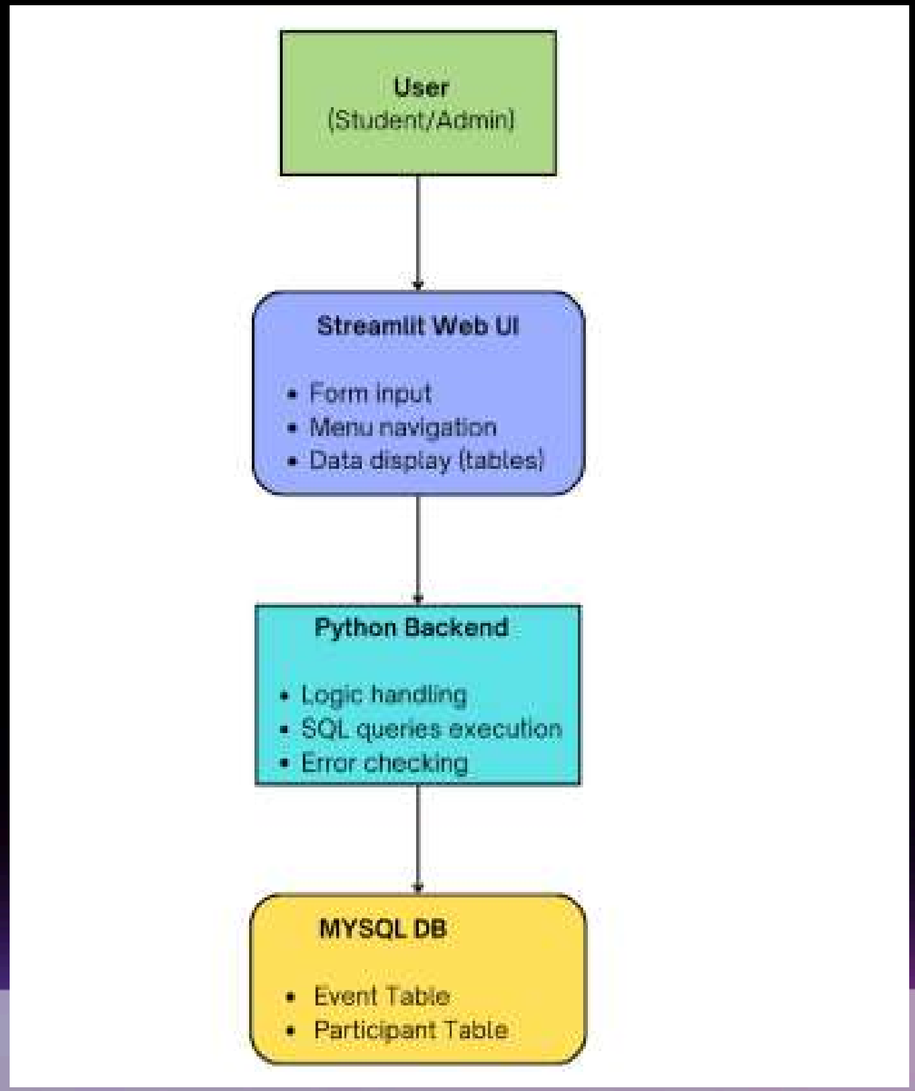
✅ 7. Location Table

Column Name	Data Type	Description
`location_id`	INT	Primary Key
`location`	VARCHAR(100)	E.g., Hall A, Online, Auditorium

ER Diagram


Admin
admin_id (PK)
username
password
Events
event_id (PK)
event_name
event_type
location
event_date
fee
capacity
description
1
N
Participants
participant_id (PK)
name
mobile
email
branch
N
Registrations
registration_id (PK)
participant_id (FK)
event_id (FK)
timestamp

System Architecture



Implementation Details

Technical Stack Used:

- Frontend: Streamlit (Python)
- Backend: Python logic for validations and data handling
- Database: MySQL
- Deployment: Runs locally using streamlit run app.py

User Functionalities

- Register/Login using mobile & email.
- View available events.
- Register for selected events
- Get confirmation messages after successful registration

Admin Functionalities

- Add/delete events (in code section)
- View registered participants
- Monitor event status (capacity, registrations)

Limitations

- Lack of Online Access – Only works on local machines.
- No Payment Integration – No UPI or online fee collection.
- No Email/SMS Notifications – No automated communication.
- Basic Security – Login lacks encryption or OTP.
- No Role-Based Access – Only single admin access; no user roles.

Impact of the Solution

For Users:

- No need to contact event coordinators manually – just log in, search, and register.
- Saves time and avoids confusion through a simple interface.
- Personalized dashboard shows their registered events.

For Admins:

- Events are created once in the code/database – no extra UI needed.
- No manual entry for participants; all data is stored and tracked in the database.

Technically:

- Authentication system ensures secure access.
- MySQL allows fast queries and safe storage of event/user data.
- Streamlit provides real-time UI updates after any action.

Future Enhancements

- Email Confirmation System:

Notify users when they register or unregister for an event.

- Event Search Filters:

Filter events by type, location, or date for better user experience.

- Live Participant Count:

Show number of spots left in real time while browsing events.

- Feedback & Event Ratings:

After attending, users can rate or review events for future improvements.

Conclusion

This project effectively solves the problem of event participation by providing a streamlined and user-friendly solution. Users simply register or log in, explore events, and join with one click – no paperwork or long forms needed.

Event organizers manage their events in the backend, reducing the need for frontend complexity while keeping full control over event creation.

Using Python, Streamlit, and MySQL, this system proves that a simple design with clear roles (admin/user) can produce an efficient, scalable, and professional event management platform.

Result Demonstration

We have implemented the project and will now demonstrate it.

REFERENCES

- Smith, A. (2020). Automating Event Management Tasks: A New Era for Organizers. Journal of Event Technology.
- Lee, B. (2019). Using SQL for Effective Data Management in Event Systems. Journal of Data Management.
- Johnson, C. (2021). Developing Event Management Systems with Python. International Journal of Programming and Web Development.
- Harris, D. (2021). The Role of Python in Building Event Management Solutions. Programming and Application Development Journal.
- Taylor, I. (2021). User Experience and Interface Design in Event Systems. Journal of Human-Computer Interaction.

By: Hamza Ali Ahmed 2023UG000142
Preethi Verma 2023UG000126
Sakshi Singh 2023UG000170
Adarsh Kumar Singh 2023UG000180

THANK YOU!