The Superior University

📝 Database Lab Project Documentation

1. Project Title

Event Management System

---

2. Group Members

List all team members with roll numbers:

- Muhammad Sohaib (067)
- Ahmad Javed (055)
- Zubair Jafar(079)
- M.Talha Malik(085)

---

3. Project Overview

The **Event Management System** is a web-based application designed to streamline the organization and management of events. It allows users to create, update, and manage events, book venues, and handle event-related data efficiently. The project leverages Flask (a Python web framework) and MySQL for backend logic and data storage. Users can interact with the system through a web interface, enabling functionality such as event creation, user and venue management, and viewing event listings.

This system solves the problem of manual event coordination by providing a centralized, automated solution for event planning, booking, and management.

---

4. Functional Requirements

The system supports the following core features and operations:

- **User Registration and Login**
  Users can sign up and log in to the system (if authentication is implemented).

- **Event Management**
  Users can create new events, view a list of existing events, update event details, and delete events.

- **Venue Management**
  Users can add new venues, update venue information, and assign events to specific venues.

- **Booking Functionality**
  The system allows booking of venues for events, ensuring that schedules do not conflict.

- **Data Display and Search**
  Events and venues can be listed, filtered, and searched for quick access.

- **Error Handling**
  Proper feedback is provided for invalid input, failed operations, or system errors.

---

5. Tools and Technologies Used

The Event Management System utilizes the following tools, libraries, and technologies:

- **Python**
  The primary programming language used for backend logic.

- **Flask**
  A lightweight Python web framework used to build the web application.

- **MySQL**
  The relational database management system used to store and manage data.

- **HTML/CSS (with Jinja2 Templating)**
  Used for creating dynamic web pages and forms.

- **SQL (DDL, DML, Views, Procedures)**
  Used extensively for defining the schema, inserting data, creating views, and stored procedures.

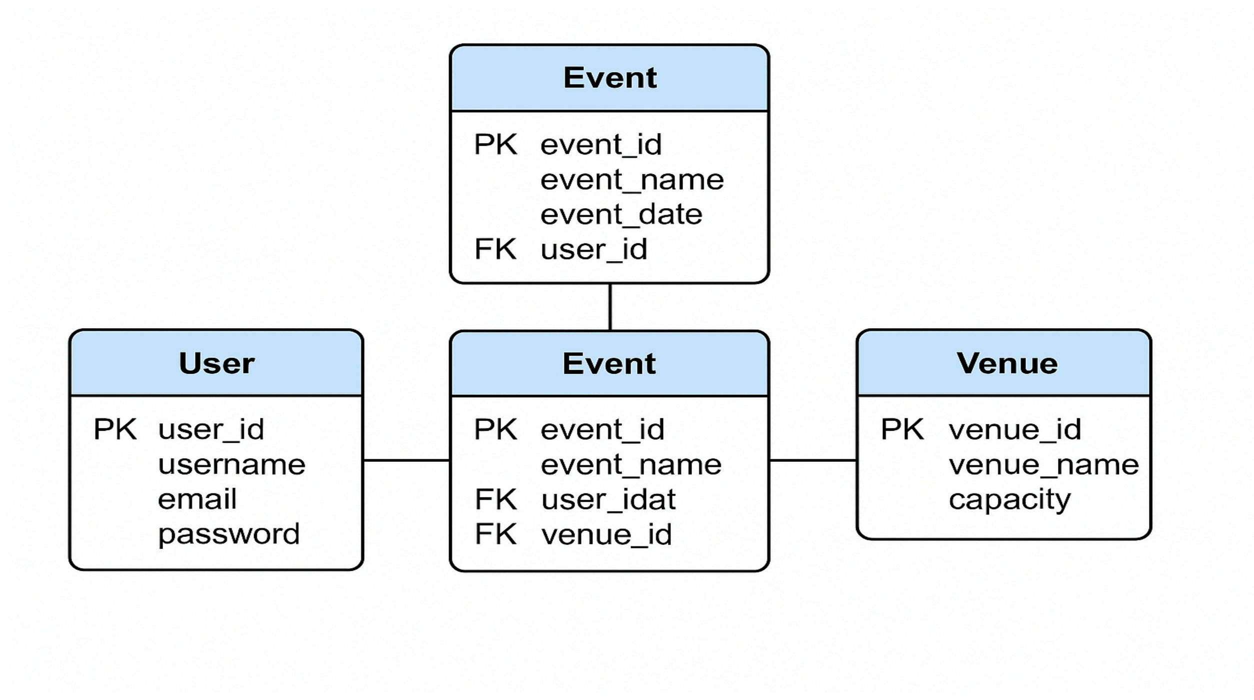
- **Git & GitHub**
  Version control system and repository hosting platform used for collaboration and source code management.

- **MySQL Workbench**
  Optional GUI tool for designing and managing the database.

---

6. Database Design (ERD)

Paste or describe your Entity Relationship Diagram (ERD) here.



---

7. SQL Concepts Used

Describe how and where you applied each of these database concepts:

- **DDL (Data Definition Language)**
- **DML (Data Manipulation Language)**
- **Joins**
- **Stored Procedures**
- **Scalar Function**
- **Inline Table-Valued Function**
- **Triggers (Before/After)**
- **Transaction Management (COMMIT/ROLLBACK)**
- **Select Queries with WHERE, HAVING, GROUP BY, ORDER BY**

8. Project Structure (MVC)

Explain the folder structure of your project and the role of each part.

`models/` *(represented by SQL files)*
Although not a separate folder, the data models are defined in SQL scripts:

- `schema.sql`: Table structures
- `views.sql`: Database views
- `procedures.sql`: Stored procedures for data logic
- `sample_data.sql`: Sample records for testing

`controllers/`
Contains logic that interacts with the database and processes user requests:

- `event_controller.py`: Handles event-related routes and logic
- `user_controller.py`: Handles user-related operations
- `venue_controller.py`: Handles venue-related functionality

`templates/` (Views)
Contains HTML templates rendered using Flask's Jinja2 engine, including:

- `add_event.html`: A form to add new events
- Other templates likely support displaying event and venue data

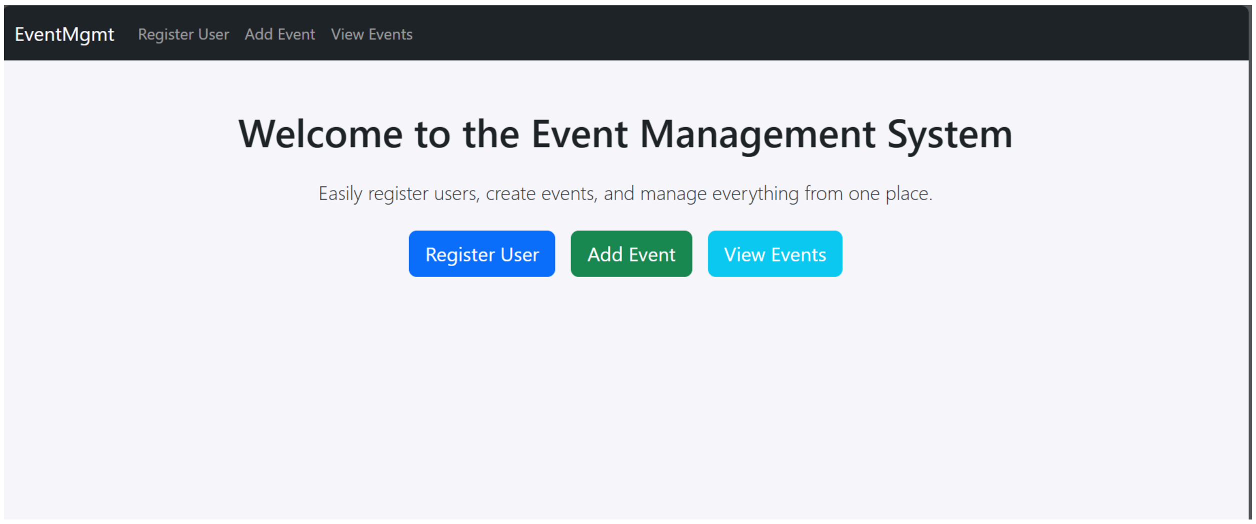
`config/` or root-level config

- `db_config.py`: Contains database connection settings
- `app.py`: Main Flask application entry point, responsible for routing and initializing the app

---

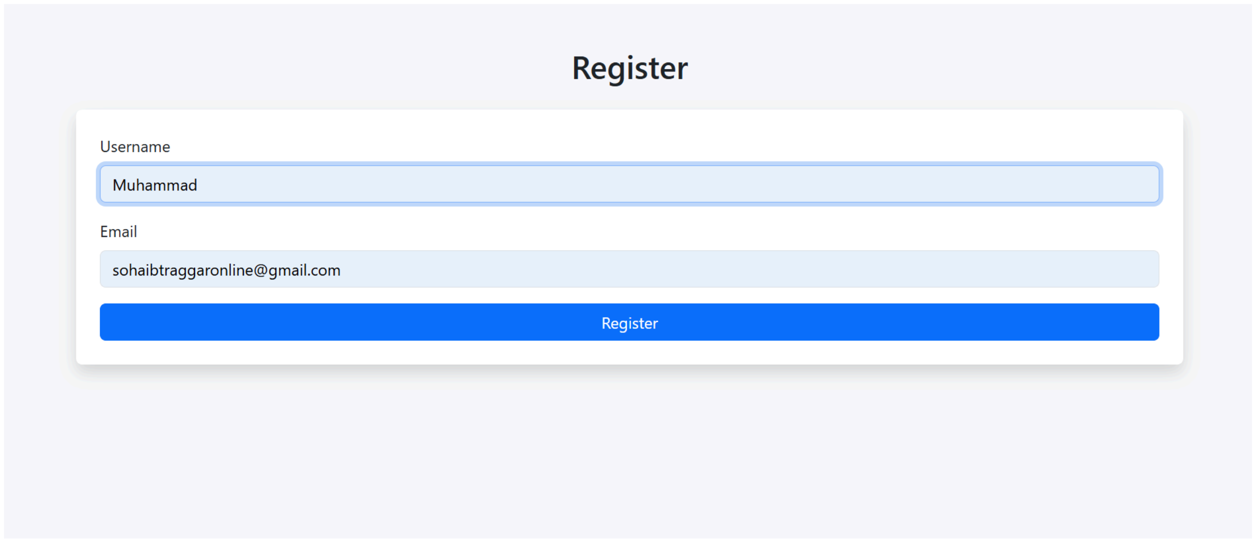
9. Screenshots of Code Output

Insert screenshots showing output for major functionalities, such as:



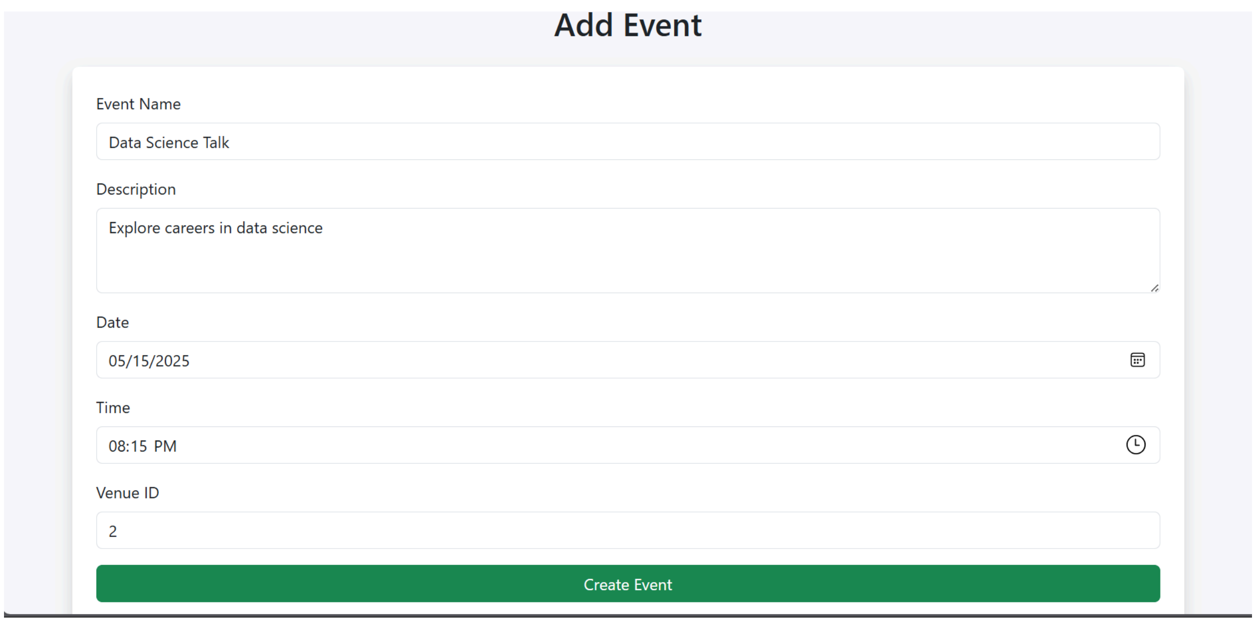
Welcome to the Event Management System

Easily register users, create events, and manage everything from one place.


Register

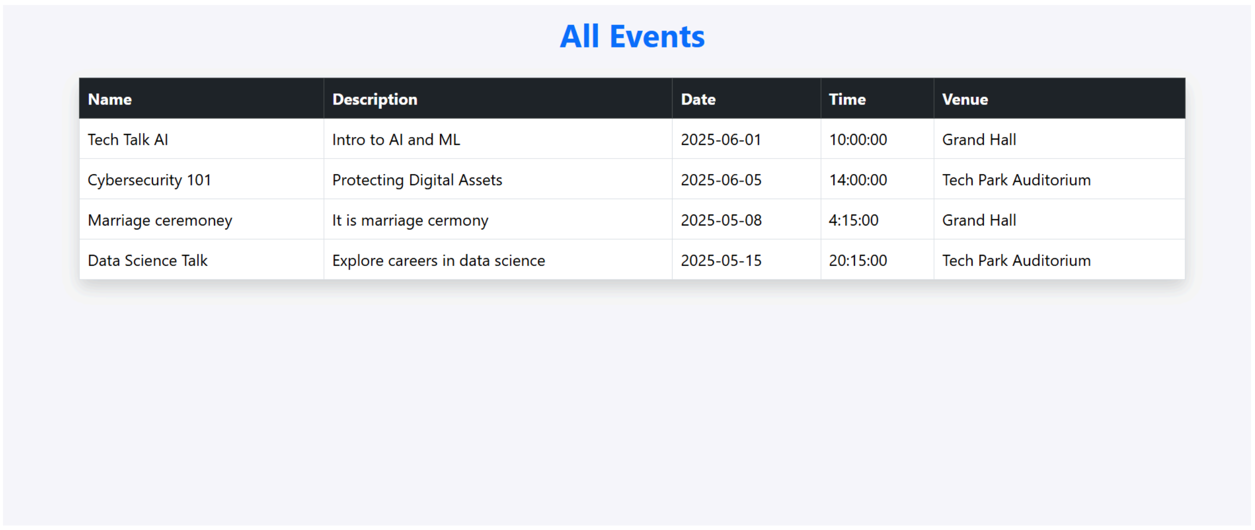
Username: Muhammad

Email: sohaibtraggaronline@gmail.com



Add Event

Event Name
Data Science Talk
Description
Explore careers in data science
Date
05/15/2025
Time
08:15 PM
Venue ID
2



All Events

Name	Description	Date	Time	Venue
Tech Talk AI	Intro to AI and ML	2025-06-01	10:00:00	Grand Hall
Cybersecurity 101	Protecting Digital Assets	2025-06-05	14:00:00	Tech Park Auditorium
Marriage ceremoney	It is marriage cermony	2025-05-08	4:15:00	Grand Hall
Data Science Talk	Explore careers in data science	2025-05-15	20:15:00	Tech Park Auditorium

---

10. Challenges Faced

Database Schema Design

Designing normalized tables and maintaining foreign key relationships between users, events, and venues was initially challenging, especially ensuring data integrity and avoiding redundancy.

Stored Procedures Integration
Writing and debugging stored procedures for operations like event creation or venue deletion required careful testing and parameter handling.

Flask Routing and Controller Logic
Managing multiple controllers for different entities and maintaining clean URL routing was complex, especially when handling form submissions and dynamic content rendering.

Template Rendering
Integrating backend data with Jinja2 templates while maintaining responsiveness and user-friendly forms took time and trial-and-error.

Error Handling and Validation
Ensuring user inputs were validated both on the frontend and backend (especially for dates, capacities, and unique entries) was critical and sometimes difficult to debug.

Lack of ERD Tool Integration
No diagram was provided with the codebase, making it harder to visualize relationships during initial analysis and testing.

---

11. Conclusion

The Event Management System successfully demonstrates the implementation of a structured, database-driven web application using Flask and MySQL. The project enabled the team to apply key database concepts such as normalization, stored procedures, and SQL views, while also practicing modular application development through the MVC architecture.

All core functionalities—such as event creation, venue management, and data display—were implemented and tested. While the project met its primary objectives, there is potential for further enhancements such as user authentication, role-based access control, calendar integrations, and more advanced error handling.

Overall, the project strengthened the team's understanding of full-stack development and database integration in a real-world context.

---

12. GitHub Repository Link

https://github.com/mstbysohaib02/Event-Management-System