CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

CASE STUDY REPORT
ON
‘EVENT MANAGEMENT SYSTEM’

Program Name: BCA

Subject Name/Code: Database Management System (23CAT-251)

Submitted by:

Name: Anshul Sharma

UID:23BCA10200

Section:23BCA4-B

Submitted to:

Name: Mr.Arvinder Singh

Designation:Professor

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

ABSTRACT:

- Introduction:
- Technique:
- System Configuration:
- INPUT:
- ER DIAGRAM:
- TABLE REALTION:
- TABULAR FORMAT:
- TABLE CREATION:
- SQL QUERIES WITH OUTPUT:
- SUMMARY:
- CONCLUSION:

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

INTRODUCTION:

An Event Management System simplifies the entire process of planning, organizing, and executing various types of events such as weddings, conferences, corporate meetings, and cultural programs. These events involve multiple activities, including booking venues, coordinating with service providers, managing clients, and handling payments.

The objective of this project is to design a structured and efficient relational database that can store, retrieve, and manage data related to multiple events and their associated requirements. It helps event organizers to keep track of client preferences, available venues, event schedules, staff assignments, and vendor services like catering or decoration.

By organizing all essential event-related data into a well-defined database system, the project ensures better accuracy, time-saving operations, and smoother workflow throughout the entire event lifecycle. This system can be used by event management companies or individuals to streamline operations, reduce manual workload, and deliver quality service to their clients.

TECHNIQUE:

The system is developed using the Relational Database Management System (RDBMS) approach, where all information is stored in well-structured tables. These tables are connected using primary and foreign keys to represent relationships between different entities such as clients, events, venues, and payments.

SQL (Structured Query Language) is used to manage the database. SQL allows us to create tables, insert data, retrieve records, update information, and delete unnecessary data. It also helps in performing advanced operations like joins and aggregations for better data analysis.

The database is normalized to reduce redundancy and maintain data accuracy. This makes the system efficient and reliable. An ER Diagram is used to visualize how the entities are related, which helps in planning and understanding the database structure before implementation.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SYSTEM CONFIGURATION:

To design and run the Event Management System, the following hardware and software configuration is recommended:

- **Operating System: Windows 10**
- **Database Software: MySQL / Oracle / PostgreSQL**
- **Programming Tools : MySQL Workbench**
- **RAM: Minimum 4 GB**
- **Processor: Intel Core i3**
- **Storage: Minimum 1 GB of available disk space**

INPUT:

The Event Management System requires various types of input data to manage events effectively. The main input fields are

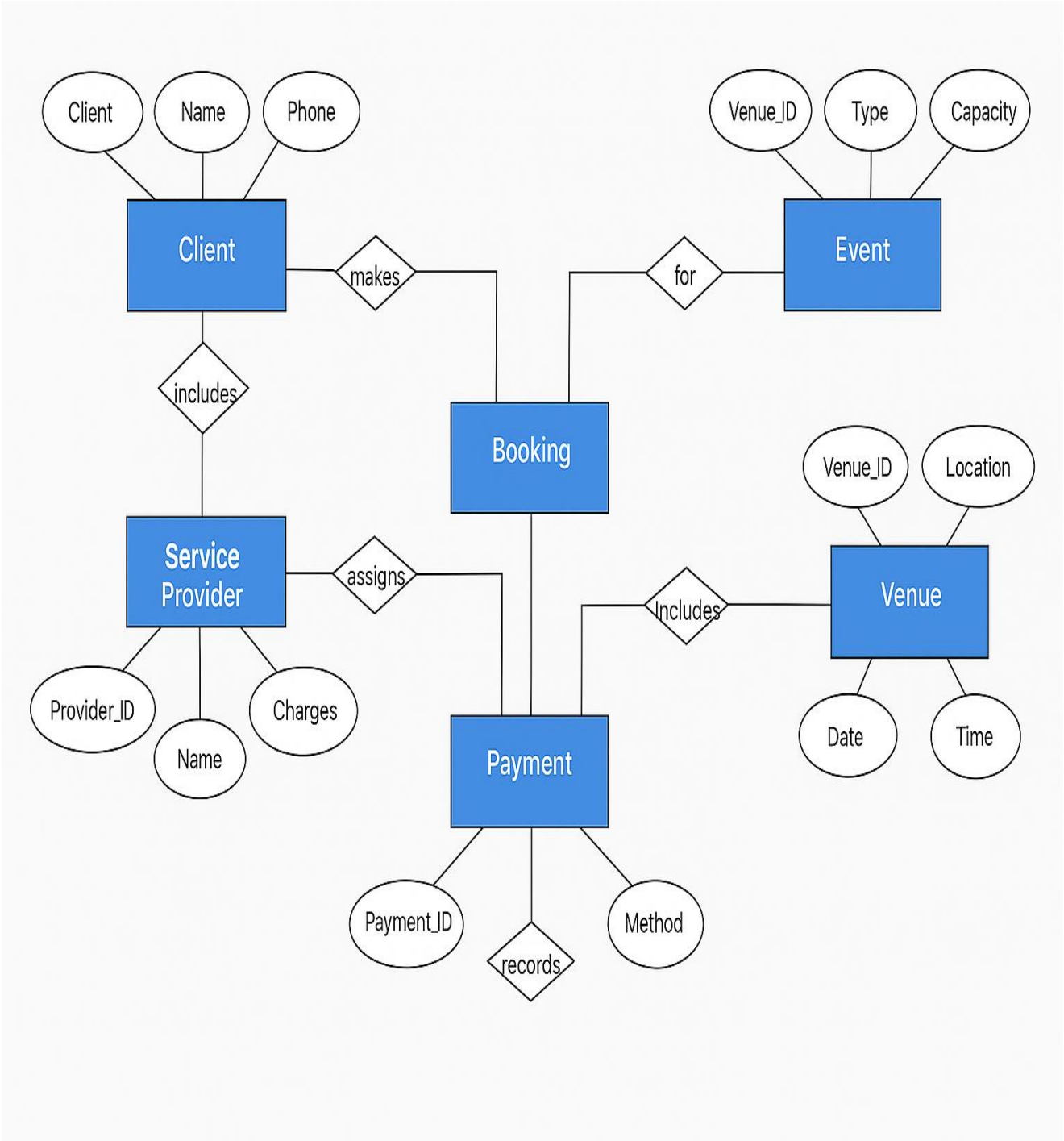
The Event Management System requires various types of input data to manage events effectively. The main input fields are:

- **Client Information:**
  - **Name**
  - **Contact Number**
  - **Email Address**
  - **Residential Address**
- **Event Details:**
  - **Event Type (e.g., Wedding, Conference, Concert, Seminar)**
  - **Event Date and Time**
  - **Number of Guests**
  - **Special Requirements (if any)**
- **Venue Information:**
  - **Venue Name**
  - **Location**
  - **Seating Capacity**
  - **Availability Status**
- **Service Provider Details:**

  - **Provider Name**
  - **Type of Service (e.g., Catering, Decoration, Photography)**
  - **Charges per Event**
  - **Contact Information**
- **Booking Details:**
  - **Client ID**
  - **Event ID**
  - **Selected Venue ID**
  - **Total Estimated Cost**
  - **Booking Date**
- **Payment Information:**
  - **Booking ID**
  - **Payment Amount**
  - **Payment Method (e.g., UPI, Card, Cash)**
  - **Payment Status (Paid/Pending)**

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

ER DIAGRAM:



CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

TABLE RELATION:

This section explains how different tables in the Event Management System database are connected using Primary Keys (PK) and Foreign Keys (FK) to maintain relational integrity between data.

1. Client → Booking

- **Relationship: One-to-Many**
- **Explanation: A single client can make multiple bookings.**
- **Key Mapping:**
  - **Client(Client) → Booking(Client) (FK)**

2. ServiceProvider → Booking

- **Relationship: One-to-Many**
- **Explanation: A service provider can be associated with many bookings.**
- **Key Mapping:**
  - **ServiceProvider(Provider_ID) → Booking(Provider_ID) (FK)**

3. Venue → Event

- **Relationship: One-to-One**
- **Explanation: Each venue is used for one event type in this model.**
- **Key Mapping:**
  - **Venue(Venue_ID) = Event(Venue_ID) (Same primary key used in both)**

4. Venue → Booking

- **Relationship: One-to-Many**
- **Explanation: One venue can have multiple bookings (on different dates).**
- **Key Mapping:**
  - **Venue(Venue_ID) → Booking(Venue_ID) (FK)**

5. Booking → Payment

- **Relationship: One-to-One**
- **Explanation: Each booking has a corresponding payment record.**
- **Key Mapping:**
  - **Booking(Booking_ID) → Payment(Booking_ID) (FK)**

6. Event → Booking *(via Venue)*

- **Relationship: Many-to-One**
- **Explanation: Multiple bookings can refer to the same event type via the venue.**
- **Key Mapping:**
  - **Booking(Venue_ID) → Event(Venue_ID) (Indirect link**

TABULAR FORMAT:

This section describes the structure of each table in the database, clearly stating the column names, data types, constraints, and detailed descriptions for all attributes.

1. Client Table:

Field	Data type	Description
Client	INT(PK)	Unique ID for each client
Name	VARCHAR(50)	Client's full name
Phone	VARCHAR(15)	Client's Phone number

Relation:

Client is linked to Booking through the foreign key Client in the Booking table.

2. Service Provider Table:

Field	Data type	Description
Provider_id	INT(P,K)	Unique ID for each service provider
Name	VARCHAR(50)	Name of the service provider
Charges	DECIMAL(10,2)	Charges or fees for services

3. Event Table:

Field	Data type	Description
Venue_id	INT(PK)	Same as Venue_ID in Venue table (linked)
Type	VARCHAR(50)	Type of event (e.g., wedding, party)
Capacity	INT	Maximum capacity of the event

Relation:

- Venue_ID in Event is connected to Venue_ID in the Venue table (1:1 relation).
- Event gives descriptive metadata about the venue setup.

4. Venue Table

Field	Data type	Description
Venue_id	INT(PK)	Unique ID for the venue
Location	VARCHAR(100)	Address or location of the venue
Date	DATE	Date of the event
Time	TIME	Time of the event

Relation:

Venue_ID is referenced by both Event and Booking tables.

5. Booking table

Field	Data type	Description
Booking_id	INT(PK)	Unique ID for each booking
Client	INT(FK)	Foreign key from Client
Venue_id	INT(FK)	Foreign key from Venue
Provider_id	INT(FK)	Foreign key from ServiceProvider

Relation:

- **Links multiple entities:**
  - **Client → identifies who made the booking**
  - **Venue_ID → where the event is being held**
  - **Provider_ID → who is servicing the event**

6.Payment Table:

Field	Data type	Description
Payment_id	INT(PK)	Unique ID for the payment
Booking_id	INT(FK)	Foreign key from Booking
Method	VARCHAR(30)	Mode of payment

Relation:

- **Booking_ID links each payment to a booking.**
- **One-to-one or one-to-many relationship depending on business logic.**

SUMMARY OF THE RELATION:

Relationship type	Description
Client → Booking	One-to-Many (A client can make multiple bookings)
Booking → Venue	Many-to-One (Many bookings can be made for a venue)
Booking → ServiceProvider	Many-to-One (Each booking involves one service provider)
Booking → Payment	One-to-One or One-to-Many (Each booking has one or more payments)
Venue → Event	One-to-One (Each venue describes one event setup)

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

TABLE CREATION:

1. Client Table:

```
CREATE TABLE Client (
Client INT PRIMARY KEY,
Name VARCHAR(50) NOT NULL,
Phone VARCHAR(15) UNIQUE
);
```

2. Service Provider Table:

```
CREATE TABLE ServiceProvider (
    Provider_ID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Charges DECIMAL(10, 2) NOT NULL
);
```

3. Venue Table:

```
CREATE TABLE Venue (
    Venue_ID INT PRIMARY KEY,
    Location VARCHAR(100) NOT NULL,
    Date DATE NOT NULL,
```

```
    Time TIME NOT NULL
);
```

4. Event table:

```
CREATE TABLE Event (
    Venue_ID INT PRIMARY KEY,
    Type VARCHAR(50) NOT NULL,
    Capacity INT NOT NULL,
    FOREIGN KEY (Venue_ID) REFERENCES Venue(Venue_ID)
);
```

5.Booking Table:

```
CREATE TABLE Booking (
    Booking_ID INT PRIMARY KEY,
    Client INT,
    Venue_ID INT,
    Provider_ID INT,
    FOREIGN KEY (Client) REFERENCES Client(Client),
    FOREIGN KEY (Venue_ID) REFERENCES Venue(Venue_ID),
    FOREIGN KEY (Provider_ID) REFERENCES
ServiceProvider(Provider_ID)
);
```

6. Payment Table:

```
CREATE TABLE Payment (
    Payment_ID INT PRIMARY KEY,
    Booking_ID INT,
    Method VARCHAR(30) NOT NULL,
    FOREIGN KEY (Booking_ID) REFERENCES Booking(Booking_ID)
);
```

These queries will create the complete database schema with all necessary primary key–foreign key relationships, data types, and constraints.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SQL QUERIES:

1. Display all clients:

```
SELECT * FROM Client;
```

Output:

Client	Name	Phone
101	Raj Verma	9876543210
102	Anjali Rao	9123456789

2.List all available venues

```
SELECT * FROM Venue;
```

Output:

Venue_id	Location	Date	Time
201	Mumbai hall	2025-04-15	18:00:00
202	Delhi Banquet	2025-05-10	17:00:00

3. Service providers with charges > ₹5000:

```
SELECT * FROM ServiceProvider WHERE Charges > 5000;
```

Output:

Provider_ID	Name	Charges
301	EventXperts	7000.00
302	PartyPros	8000.00

4. All bookings with client names:

```
SELECT B.Booking_ID, C.Name AS Client_Name, V.Location
FROM Booking B
JOIN Client C ON B.Client = C.Client
JOIN Venue V ON B.Venue_ID = V.Venue_ID;
```

Output:

Booking_id	Client_name	Location
401	Raj Verma	Mumbai Hall
402	Anjali Rao	Delhi Banquet

5. Events with capacity > 200:

```
SELECT * FROM Event WHERE Capacity > 200;
```

Output:

Venue_id	Type	Capacity
201	Wedding	300

6. Number of bookings per client:

```
SELECT Client, COUNT(*) AS Total_Bookings

FROM Booking

GROUP BY Client;
```

Client	Total_Bookings
101	1
102	2

7. Bookings with payment method:

```
SELECT B.Booking_ID, P.Method

FROM Booking B

JOIN Payment P ON B.Booking_ID = P.Booking_ID;
```

Output:

Booking_id	Method
401	UPI
402	Cash

8. Total revenue by service provider:

```
SELECT SP.Name, SUM(SP.Charges) AS Total_Revenue

FROM Booking B

JOIN ServiceProvider SP ON B.Provider_ID = SP.Provider_ID

GROUP BY SP.Name;
```

Output:

Name	Total_revenue
EventXperts	7000.00
PartyPros	8000.00

9. Clients who paid via UPI:

```
SELECT DISTINCT C.Name

FROM Client C

JOIN Booking B ON C.Client = B.Client

JOIN Payment P ON B.Booking_ID = P.Booking_ID

WHERE P.Method = 'UPI';
```

Output:

Name

Raj Verma

10. Venue location for each event:

```
SELECT E.Type, V.Location

FROM Event E

JOIN Venue V ON E.Venue_ID = V.Venue_ID;
```

Output:

Type	Location
Wedding	Mumbai Hall
Birthday	Delhi Banquet

11. Bookings on 2025-04-15:

```
SELECT B.Booking_ID, V.Date, C.Name
FROM Booking B
JOIN Venue V ON B.Venue_ID = V.Venue_ID
JOIN Client C ON B.Client = C.Client
WHERE V.Date = '2025-04-15';
```

Output:

Booking_id	Date	Name
401	2025-04-15	Raj Verma

12. Events sorted by capacity:

```
SELECT * FROM Event ORDER BY Capacity DESC;
```

Output:

Venue_id	Type	Capacity
201	Wedding	300
202	Birthday	100

13. Most expensive service provider:

```
SELECT * FROM ServiceProvider

WHERE Charges = (SELECT MAX(Charges) FROM
ServiceProvider);
```

Output:

Provider_id	Name	Charges
302	PartyPros	8000.00

14. Bookings count per venue:

```
SELECT Venue_ID, COUNT(*) AS Booking_Count

FROM Booking

GROUP BY Venue_ID;
```

Output:

Venue_id	Booking_Count
201	1
202	1

15. Clients with payment methods:

```
SELECT C.Name, P.Method

FROM Client C

LEFT JOIN Booking B ON C.Client = B.Client

LEFT JOIN Payment P ON B.Booking_ID = P.Booking_ID;
```

Output:

Name	Method
Raj Verma	UPI
Anjali Rao	Cash

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SUMMARY:

The Event Management System is a comprehensive and user-friendly database solution designed to simplify the organization, planning, and management of events. This system integrates essential entities such as Clients, Service Providers, Venues, Events, Bookings, and Payments into a well-structured relational database.

The system enables users to manage and track various events—such as weddings, birthdays, and corporate functions—by allowing bookings of venues, assigning service providers, and recording payments. The ER diagram effectively illustrates the relationships between these entities, showing how each event is linked to a client, venue, and service provider through a centralized booking table.

Advanced SQL queries were implemented to retrieve real-time data and insights, including client details, service provider charges, venue schedules, event capacities, booking statuses, and revenue analysis. Each table was created with appropriate constraints such as primary keys and foreign keys to maintain data integrity.

By using this database-driven approach, the system minimizes manual errors, speeds up event coordination, and enhances overall operational efficiency. It serves as a reliable backend for any future web or app-based event management platform.

CONCLUSION:

The Event Management System project successfully demonstrates the practical application of database concepts in real-life scenarios. By using a well-structured relational database model, we have efficiently handled key components of event management such as client records, venue bookings, service provider coordination, and payment tracking.

Through the use of Entity-Relationship Diagrams, proper table creation, and SQL queries, the system ensures smooth data flow, minimal redundancy, and accurate reporting. Complex queries were used to extract meaningful insights, proving the robustness and flexibility of the database design.

This project highlights how a database management system can simplify and automate the operations of an event management company, ensuring efficient handling of multiple events, reducing manual work, and improving customer satisfaction. It lays a strong foundation for future expansion into a fully functional web or mobile application.

Overall, the project meets its objectives and showcases the importance of DBMS in building scalable and organized information systems.