Event Management System

Project Report

By
Shweta Mestry – AF04970116
Pranali Hivale – AF04970197

Index

Acknowledgement

The project **"Event Management System"** is the Project work carried out by

Name	Enrollment No
Shweta Mestry	**AF04970116**
Pranali Hivale	**AF04970197**

We are thankful to our project guide for guiding us to complete the Project.

Their suggestions and valuable information regarding the formation of the Project Report have provided us a lot of help in completing the Project and its related topics.

Abstract

Managing events efficiently requires seamless coordination among organizers, venues, attendees, and vendors — a task that often becomes complex without a structured system. The Event Management System is a console-based Java application developed using Hibernate ORM and MySQL, designed to streamline the entire event planning and management process. The system enables users to add, update, view, and delete details of events, organizers, venues, vendors, and attendees, while maintaining relationships such as event scheduling, venue allocation, and service management. Through its modular design and database-driven architecture, the application ensures data integrity, reduces manual workload, and enhances operational efficiency. By integrating all key aspects of event coordination into a unified platform, the Event Management System provides a reliable, scalable, and user-friendly solution for organizing and managing events effectively.

1. Introduction

Managing events efficiently is a crucial aspect of organizational, corporate, and social success. In an era where events range from small gatherings to large-scale conferences, individuals and organizations often face challenges in coordinating multiple aspects such as venue booking, attendee management, vendor coordination, and scheduling. Manual methods of event management—such as maintaining spreadsheets or written records—are time-consuming, error-prone, and lack the scalability required for modern event operations. To overcome these limitations, technology-driven solutions can play a vital role in streamlining and automating event management processes.
The Event Management System Is a console-based application developed using Java, Hibernate ORM, and MySQL, designed to simplify and automate the core activities of event planning and coordination. The system provides modules for managing Organizers, Venues, Events, Attendees, and Vendors, ensuring smooth data flow and relationship handling through a relational database.

1. Background of the Study

Event management plays a vital role across various domains including education, entertainment, business, and public administration. Whether it's a college fest, wedding, product launch, or conference, successful event organization requires careful planning, coordination, and communication among multiple stakeholders. Traditional approaches rely heavily on manual documentation, phone calls, and emails, often leading to scheduling conflicts, data redundancy, and mismanagement of resources.
With advancements in software technologies, digital systems now provide automated solutions that enhance coordination, minimize human errors, and improve productivity. The Event Management System is designed with this vision — to serve as a unified digital platform that stores, retrieves, and manages all event-related information efficiently

2. Motivation

The motivation behind developing this project arises from the growing demand for a structured, reliable, and easy-to-use system that can handle multiple components of event management in one place. The challenges observed in traditional event planning include:

- Manual errors and duplication in managing organizer, vendor, and attendee data.
- Inefficient communication between stakeholders leading to last-minute issues.
- Difficulty in tracking venues, service providers, and event logistics.
- Lack of centralized records, making data retrieval and updates cumbersome.

This project addresses these challenges by leveraging Java's object-oriented design and Hibernate's ORM capabilities to build a modular and scalable system that automates repetitive tasks and ensures smooth event execution.

3. Significance of the Study

This project holds significance for multiple stakeholders:

For Event Organizers: Provides an efficient tool for planning, managing, and tracking multiple events.

For Venues and Vendors: Ensures proper allocation and scheduling to avoid conflicts.

For Attendees: Enhances coordination and participation management.

For Developers and Students: Serves as a practical example of integrating Java, Hibernate, and MySQL for real-world applications.

By introducing automation into event management, the system reduces manual workload, increases data accuracy, and ensures that all event operations are conducted smoothly.

4. Features of Event Management System:

The **Event Management System** offers multiple modules that make event organization and coordination efficient, structured, and error-free.

Organizer Management

a) Add, update, view, and delete organizer details such as name, address, contact, and email.
b) Assign multiple events to a single organizer.
c) Maintain a centralized database of all registered organizers.

Venue Management

a) Add and manage venue details like name, address, capacity, owner, and contact information.
b) Allocate venues to specific events based on availability.
c) Prevent double-booking and ensure resource optimization.

Attendee Management

a) Register attendees with their name, email, phone, and address.
b) Assign attendees to specific events through many-to-many relationships.
c) View, update, or remove attendees as per event requirements.

Vendor Management

a) Register vendors providing services such as catering, decoration, sound, or lighting.
b) Associate multiple vendors with one or more events.
c) Maintain vendor contact details for easy communication and coordination.

Event Management:

a) Create, update, and delete events.
b) Assign organizers, venues, vendors, and attendees to events.

c) Generate event details and summary reports.

5. Reporting and Analytics:

- Generate reports of past and upcoming events.
- Provide summaries of attendees, vendors, and venue usage.

2. System Analysis

System analysis is a crucial phase in the Software Development Life Cycle (SDLC). It involves identifying the existing challenges in event coordination, analysing functional requirements, and evaluating the feasibility of the proposed solution. For the Event Management System, this stage focuses on understanding the needs of organizers, venues, vendors, and attendees, and justifying the development of a centralized, automated platform to streamline event-related operations.

1. Problem Definition

Managing finances is a significant challenge faced by individuals in today's society. Managing events manually poses several challenges in terms of coordination, communication, and record-keeping. Traditional methods such as maintaining spreadsheets, handwritten notes, or standalone tools have significant drawbacks, including:

Data Duplication and Inconsistency: Event details, attendee lists, and vendor information are often stored separately, leading to confusion and data mismatches.

Poor Coordination: Manual systems make it difficult to synchronize tasks among organizers, vendors, and participants.

Lack of Centralized Information: Event-related data like venues, attendees, and schedules are scattered across multiple sources, making management inefficient.

2. Objectives of the System

The proposed Event Management System aims to:

- Provide a centralized platform to manage all aspects of event organization, including organizers, venues, attendees, vendors, and events.
- Enable users to create, update, view, and delete event details efficiently through a structured database system.
- Facilitate better coordination among organizers, venues, and vendors by maintaining clear relationships and data consistency.
- Simplify event scheduling, registration, and service management to minimize manual errors and duplication.
- Ensure data integrity and reliability through database connectivity and Hibernate ORM integration.
- Provide user-friendly console-based interaction for performing CRUD operations with ease.
- Lay the foundation for future scalability, such as adding analytics, automated notifications, or a web interface for real-time event management.

3. Feasibility Study

Feasibility analysis evaluates the practicality of developing the proposed Event Management System. It ensures that the system is viable in terms of technology, cost, operations, and time.

a. Technical Feasibility

The system is developed using Java with Hibernate ORM for database interaction and MySQL as the backend database. It uses a console-based interface for CRUD operations, allowing seamless interaction with data entities such as events, organizers, venues, attendees, and vendors.

All tools and technologies required (Java JDK, Hibernate, and MySQL) are open-source and widely supported.

Conclusion: The required technologies are reliable, available, and suitable for the development environment, making the project technically feasible.

b. Economic Feasibility

The system is highly cost-effective since it utilizes open-source technologies like Java, Hibernate, and MySQL. There are no licensing or subscription costs involved.

The system reduces manual effort and enhances event coordination efficiency, offering significant long-term benefits compared to minimal development costs.

Conclusion: The project is economically feasible.

c. Operational Feasibility

The system provides a user-friendly and structured interface for managing event-related operations. Users such as organizers, vendors, and attendees can easily adapt to it. It minimizes human errors and automates repetitive tasks.

Security features like controlled access ensure safe data handling and prevent unauthorized use.

Conclusion: The system is operationally feasible.

d. Time Feasibility

The project can be developed efficiently within the allotted timeframe using an iterative approach. Each module—such as Event, Organizer, Venue, Attendee, and Vendor—can be implemented and tested separately.

The modular structure enables continuous integration and early feedback.

Conclusion: The project is time-feasible.

4. System Requirements

a. Functional Requirements:

- The Event Management System must support the following functionalities:

- Add, view, update, and delete organizers.
- Register and manage venues (name, capacity, location).

- Add, update, and delete attendees.
- Add and manage vendors for each event.
- Create, update, and delete events.
- Assign venues, organizers, vendors, and attendees to specific events.
- Generate event details and summary reports.

b. Non-Functional Requirements:

- Usability: Simple console interface for easy interaction.
- Reliability: Ensure consistent data storage and retrieval using Hibernate.
- Scalability: System can handle multiple events and participants efficiently.
- Security: Prevent unauthorized access to sensitive event data.
- Performance: CRUD operations and data retrieval should execute quickly..

c. Hardware Requirements:

- Processor: Intel i5 or higher.
- RAM: 4 GB minimum (8 GB recommended).
- Storage: 500 MB for database and reports.

d. Software Requirements

- Operating System: Windows / Linux / macOS
- Database: MySQL 8.0 or above
- Programming Language: Java 17 or higher
- Frameworks/Libraries: Hibernate ORM, Junit (for testing)
- IDE: IntelliJ IDEA / Eclipse / NetBeans
- Build Tool: Apache Maven

5. Proposed System

The proposed Event Management System provides a unified digital platform for efficiently managing all aspects of event organization.

Event Creation & Scheduling: Organizers can create and manage events with details such as event name, date, time, type, and assigned venue.

Venue Management: Venues can be registered with their name, capacity, location, and contact details, ensuring proper allocation and avoiding double bookings.

Attendee Management: Attendees can be added, listed, updated, or removed for each event, maintaining accurate participation records.

Vendor Management: Vendors offering different services (like catering, decoration, or sound) can be linked to specific events.

Data Centralization: All information related to organizers, venues, attendees, and vendors is stored in a centralized database, improving accessibility and consistency.

Reports & Insights: The system provides an overview of events, participants, and vendors, supporting efficient tracking and decision-making.

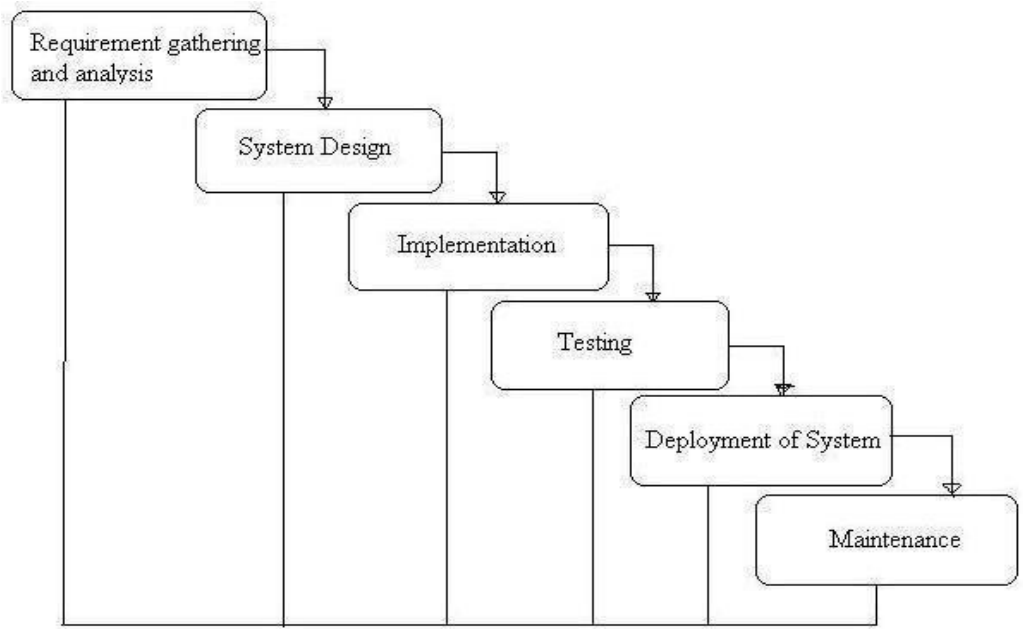

Fig. 5.2 . Phases of Development

6. Phases of Development

1. **Requirement Analysis & System Study**
    - Identifying the goals of the Event Management System and understanding existing challenges in event coordination.
    - Gathering requirements from stakeholders such as organizers, vendors, and attendees.
    - Defining functional and non-functional requirements for modules like Event, Organizer, Venue, Vendor, and Attendee.
2. **System Design**
    - Designing the database schema using MySQL to represent relationships among entities such as Event, Organizer, Venue, Attendee, and Vendor.
    - Developing the architecture to ensure modularity and scalability.
    - Planning the console-based user interaction flow for efficient CRUD operations.
3. **Implementation (Coding)**
    - Backend development using Java with Hibernate ORM for database operations.

- Integration of MySQL for persistent data storage.
- Creating entity classes and DAO (Data Access Object) layers to handle CRUD functionalities.

4. **Testing & Debugging**
   - Conducting unit testing on each module (Organizer, Event, Venue, Vendor, Attendee).
   - Performing integration testing to ensure smooth data flow across modules.
   - Debugging and refining system logic for improved reliability and accuracy.
5. **Deployment & Maintenance**
   - Deploying the application on local environments for usage and demonstration.
   - Ensuring proper configuration of the database and Hibernate settings.
   - Providing continuous maintenance for bug fixes, feature enhancements, and database optimization.

7. ER diagram:

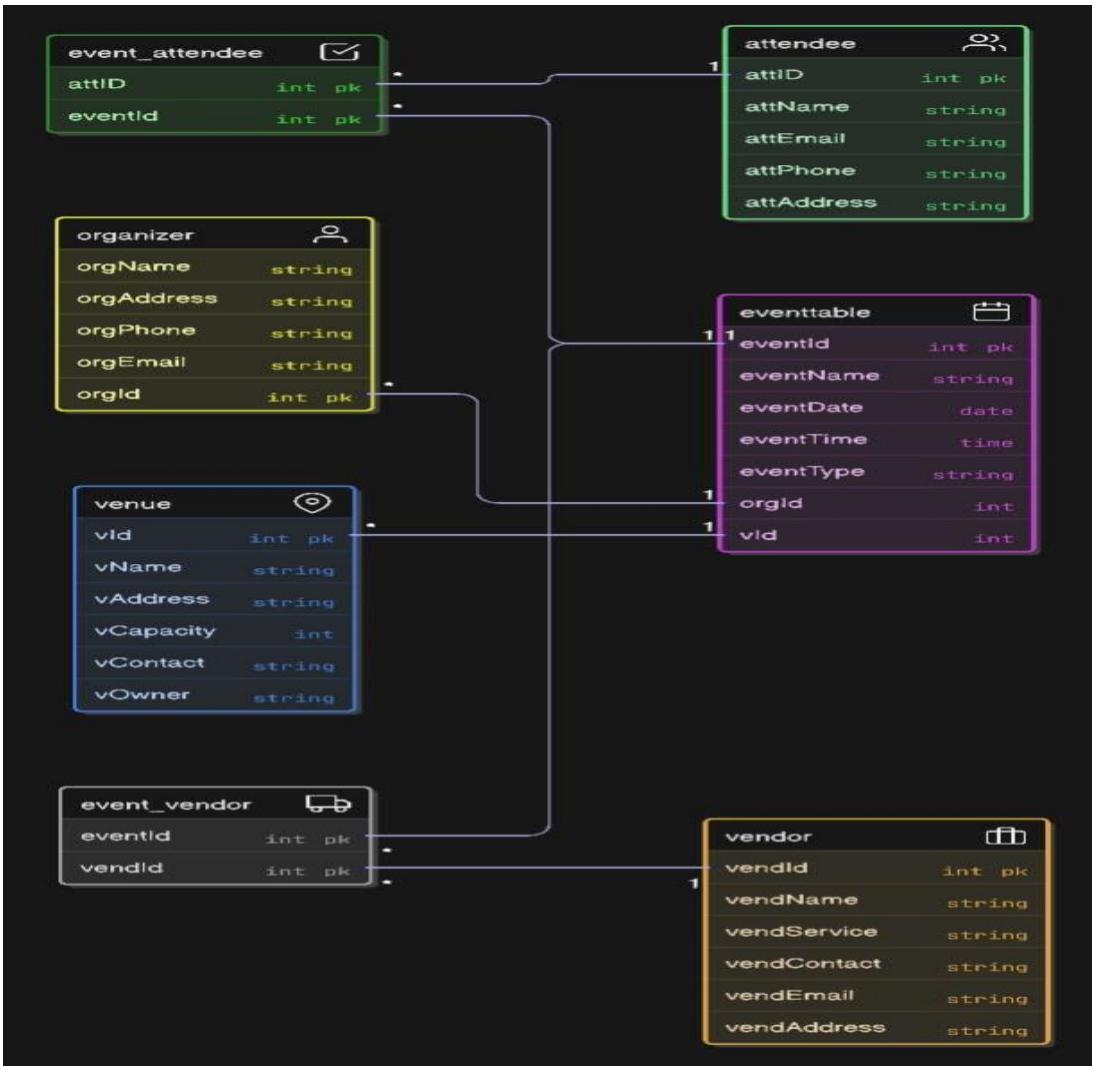

Fig. 2.7. ER Diagram

The ER diagram for the Event Management System represents the logical structure of the database, showing entities, their attributes, and relationships among them. It provides a clear view of how organizers, events, venues, attendees, and vendors interact within the system.

1. **Entities and Their Attributes**

- Organizer
  Attributes: org_id (PK), org_name, org_address, org_phone, org_email
  Represents the event organizer who is responsible for managing and conducting events.
- Venue
  Attributes: v_id (PK), v_name, v_address, v_capacity, v_contact, v_owner
  Stores details of event locations including name, capacity, and contact information.
- Attendee
  Attributes: att_id (PK), att_name, att_email, att_phone, att_address
  Represents participants who attend the events.
- Vendor
  Attributes: vend_id (PK), vend_name, vend_service, vend_contact, vend_email, vend_address
  Defines service providers who offer various event-related services like catering, decoration, or sound management.
- Event
  Attributes: event_id (PK), event_name, event_date, event_time, event_type, org_id (FK), v_id (FK)
  Represents individual events, linking organizers and venues with scheduled dates and types (e.g., cultural, corporate, wedding).
- Event_Attendee
  Attributes: event_id (FK), att_id (FK)
  Junction table that defines a many-to-many relationship between Events and Attendees.
- Event_Vendor
  Attributes: event_id (FK), vend_id (FK)
  Junction table that defines a many-to-many relationship between Events and Vendors.

2. **Relationships**

- Organizer → Event: One organizer can manage multiple events (1 : M).
- Event → Venue: Each event takes place at one venue, but a venue can host multiple events (1 : M).
- Event ↔ Attendee: Each event can have multiple attendees, and each attendee can attend multiple events (M : M) — handled by the Event_Attendee table.
- Event ↔ Vendor: Each event can have multiple vendors, and each vendor can serve

multiple events (M : M) — handled by the Event_Vendor table.

3. **ER Diagram Summary**

- The central entity in the system is the Event, which connects to Organizers, Venues, Attendees, and Vendors.
- Many-to-many relationships between Events and both Attendees and Vendors ensure flexibility and scalability.
- The database structure is normalized to avoid redundancy and ensure data integrity.
- This ER design supports efficient event coordination, reporting, and data management across all modules.

3. System design

A) Modules

1. Organizer Module

This module manages all functionalities related to event organizers. Organizers are responsible for planning and coordinating events.

They can:

- Add, view, update, and delete organizer information.
- Manage events under their supervision.
- Assign venues, vendors, and attendees to their respective events.
- View a summary of all events organized by them.

2. Event Module

This is the central module of the system and handles all event-related activities.

Users can:

- Create new events with details such as name, type, date, and time.
- Assign an organizer and venue to each event.
- Add or remove attendees and vendors associated with an event.
- View detailed event information, including venue, vendor services, and attendee lists.
- Update or delete events as required.

3. Venue Module

The venue module stores and manages information about event locations.

Users can:

- Add, update, and delete venue details (name, address, capacity, contact).
- Assign venues to events based on availability.
- View all registered venues and their usage in events.

4.Attendee Module

This module maintains attendee records and their participation in various events.

Users can:

- Add new attendees with name, email, phone, and address details.
- Update or delete attendee information.
- Associate attendees with events they are participating in.
- View all attendees linked to specific events.

5.Vendor Module

The vendor module manages service providers who assist in event operations.

Users can:

- Add, update, and delete vendor information.

- Assign vendors to events for services like catering, decoration, or logistics.
- View all vendors and their associated events.

6.System Module (Backend Services)
This module forms the backbone of the Event Management System. It handles the internal logic, data flow, and database interaction.
It includes:

- Performing CRUD operations on all entities (Organizer, Venue, Attendee, Vendor, Event).
- Managing many-to-many relationships (Event–Attendee, Event–Vendor).
- Handling data persistence using Hibernate ORM.
- Managing session creation, transaction handling, and database connectivity.
- Ensuring data consistency and integrity across all modules.

B) Data Structure of All Modules

We have organized one database named event_management for the system design.
It is implemented using MySQL and consists of several customized tables representing the entities and relationships involved in the event management process.

1.Customized Tables
Organizer Table (organizer)
Stores information about event organizers.
Attributes: org_id, org_name, org_address, org_phone, org_email

Venue Table (venue)
Stores details of event venues.
Attributes: v_id, v_name, v_address, v_capacity, v_contact, v_owner

Attendee Table (attendee)
Stores information about participants attending events.
Attributes: att_id, att_name, att_email, att_phone, att_address

Vendor Table (vendor)
Stores vendor details and their services.
Attributes: vend_id, vend_name, vend_service, vend_contact, vend_email, vend_address

Event Table (event)
Stores all event details, linking organizers and venues.
Attributes: event_id, event_name, event_date, event_time, event_type, org_id, v_id

Foreign Keys:

Org_id → References organizer(org_id)
V_id → References venue(v_id)
Event_Attendee Table (event_attendee)
Defines the many-to-many relationship between Events and Attendees.
Attributes: event_id, att_id

Foreign Keys:
Event_id → References event(event_id)
Att_id → References attendee(att_id)
Event_Vendor Table (event_vendor)
Defines the many-to-many relationship between Events and Vendors.
Attributes: event_id, vend_id

Foreign Keys:
Event_id → References event(event_id)
Vend_id → References vendor(vend_id)

2.Relationships Between Tables (ER/Class Diagram)

Organizer → Event: One organizer can manage multiple events (1 : M).
Event → Venue: Each event is held at one venue, but a venue can host many events (1 : M).
Event ↔ Attendee: Each event can have multiple attendees, and each attendee can attend multiple events (M : M) — handled through event_attendee.
Event ↔ Vendor: Each event can have multiple vendors, and each vendor can serve multiple events (M : M) — handled through event_vendor.

This structure ensures data normalization, eliminates redundancy, and maintains referential integrity across all modules. It supports scalable and efficient management of all event-related data in the system.

C) Screenshots

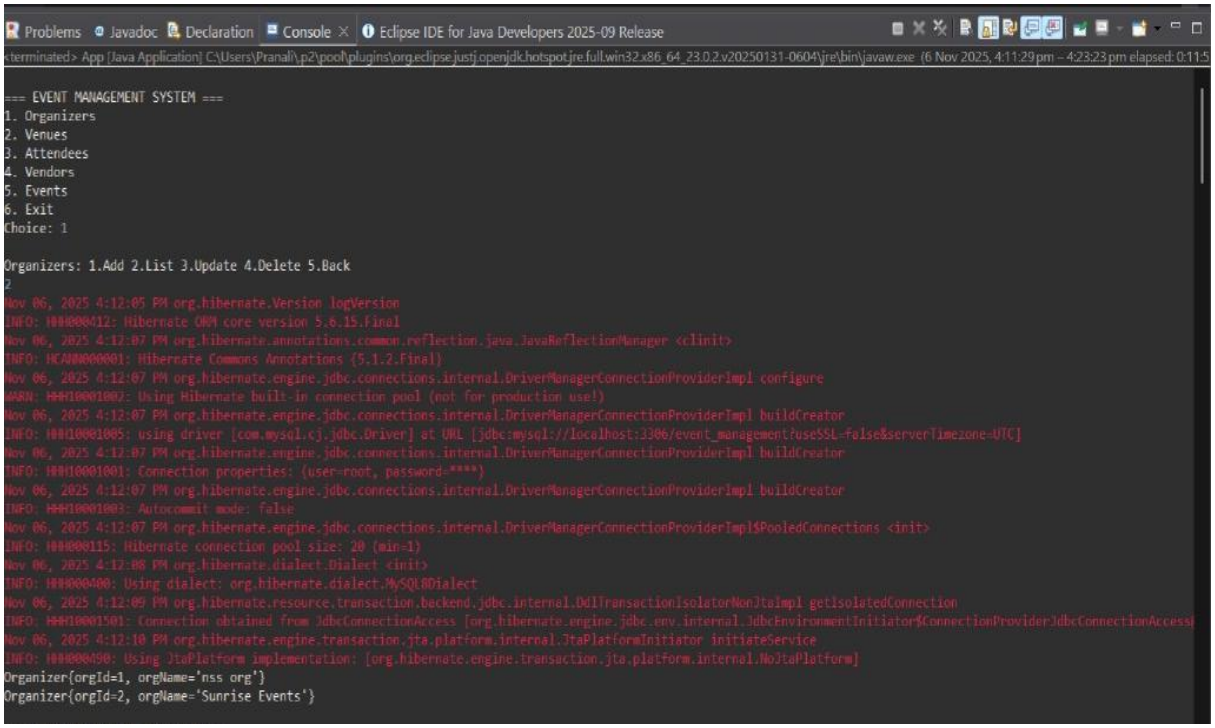

Fig. 3.1. List of option displayed on screen

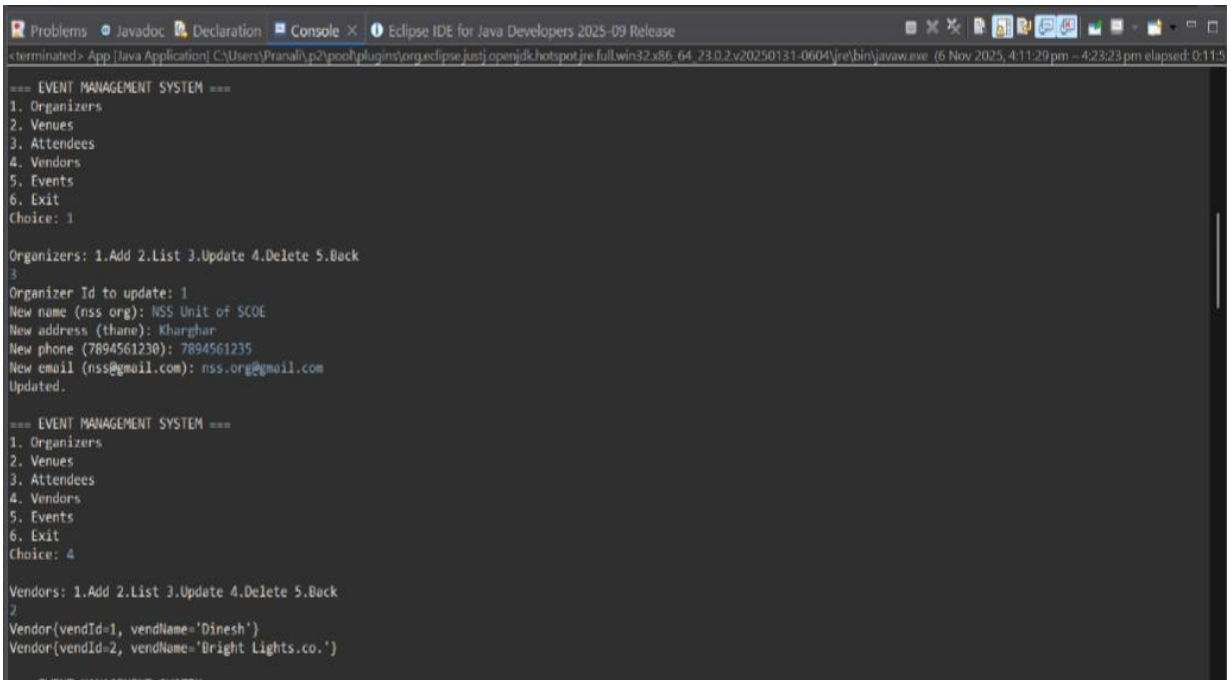

Fig. 3.2. Updated Organizer

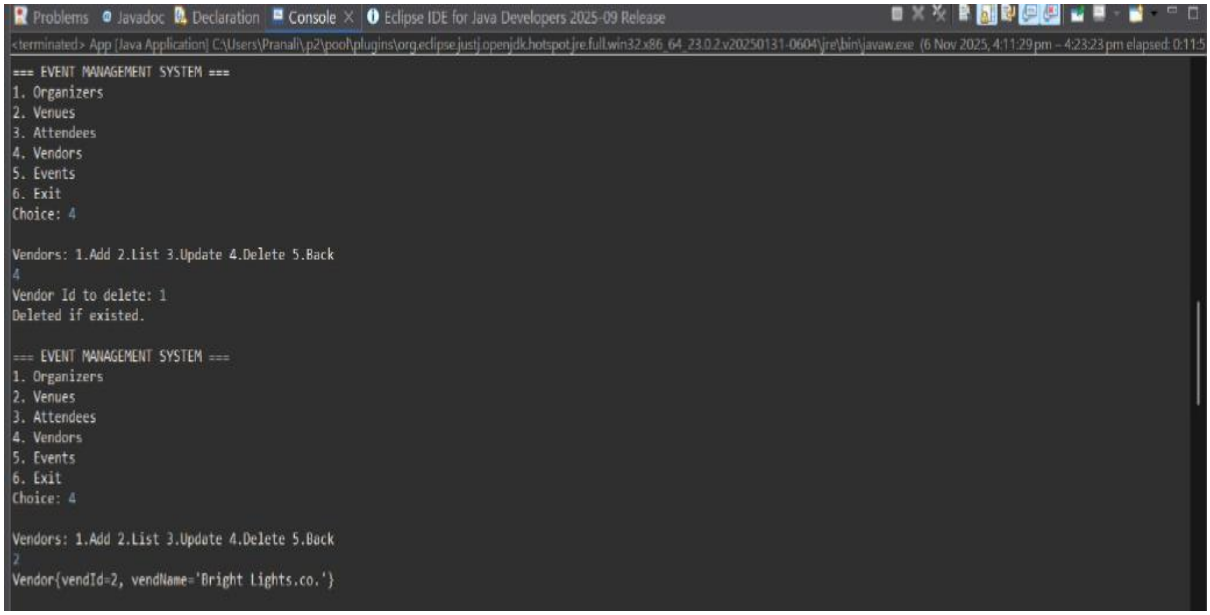

Fig. 3.3. Vendor delete and list operation

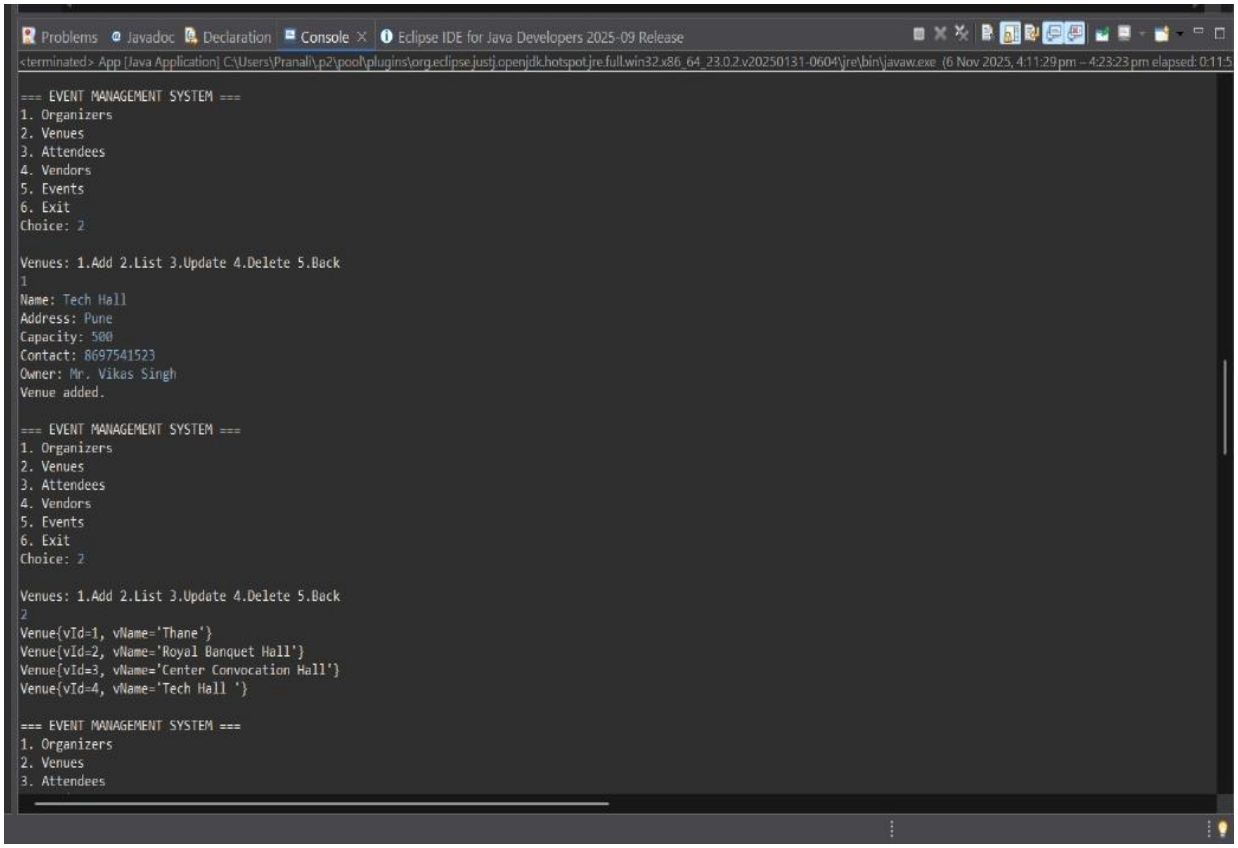

Fig. 3.4. Venue add and list operation

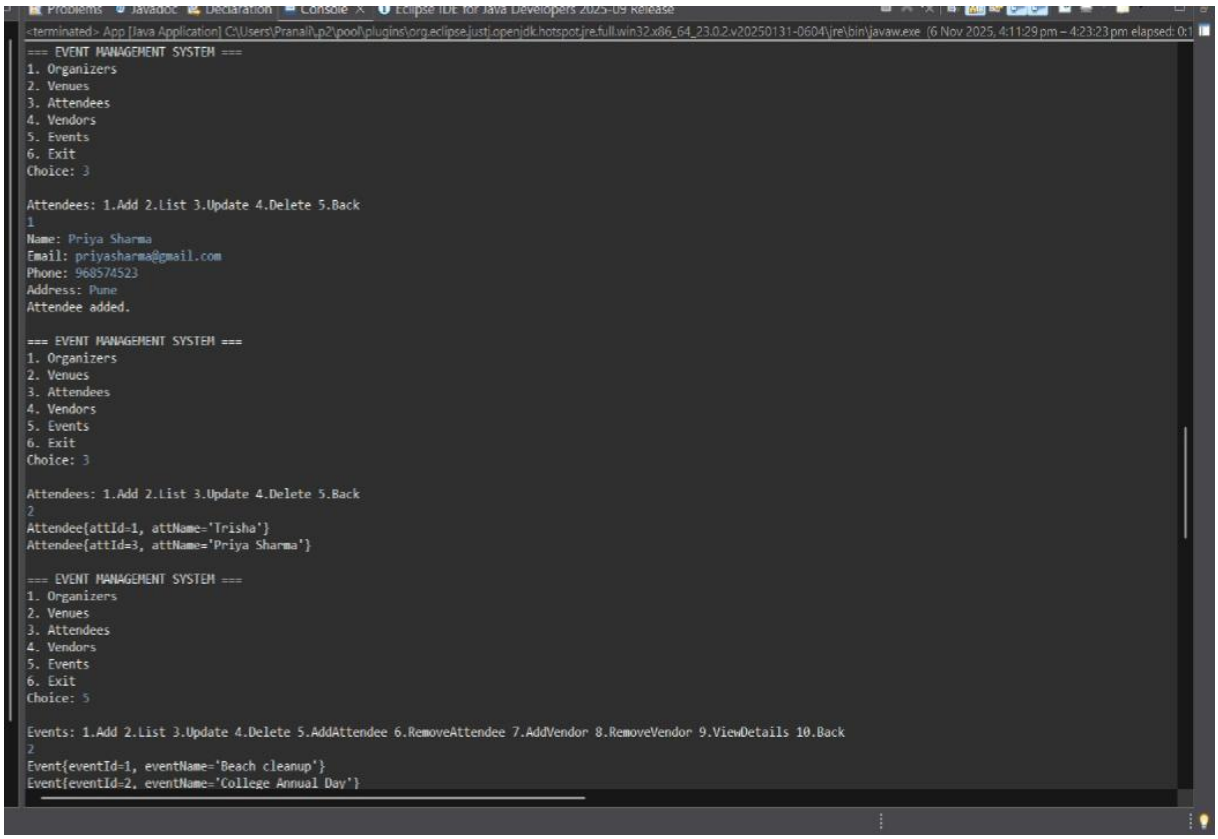

Fig. 3.5. Attendee add and list operation and event list operation

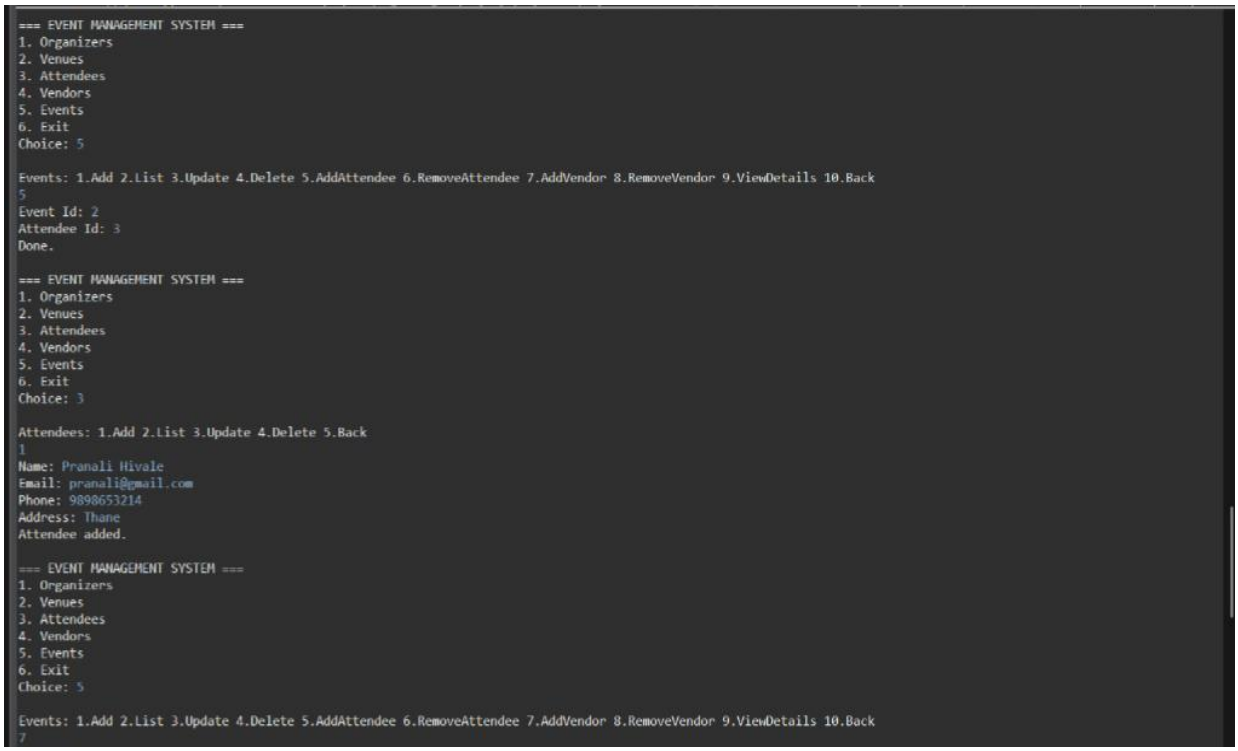

Fig. 3.6. Event curd operations

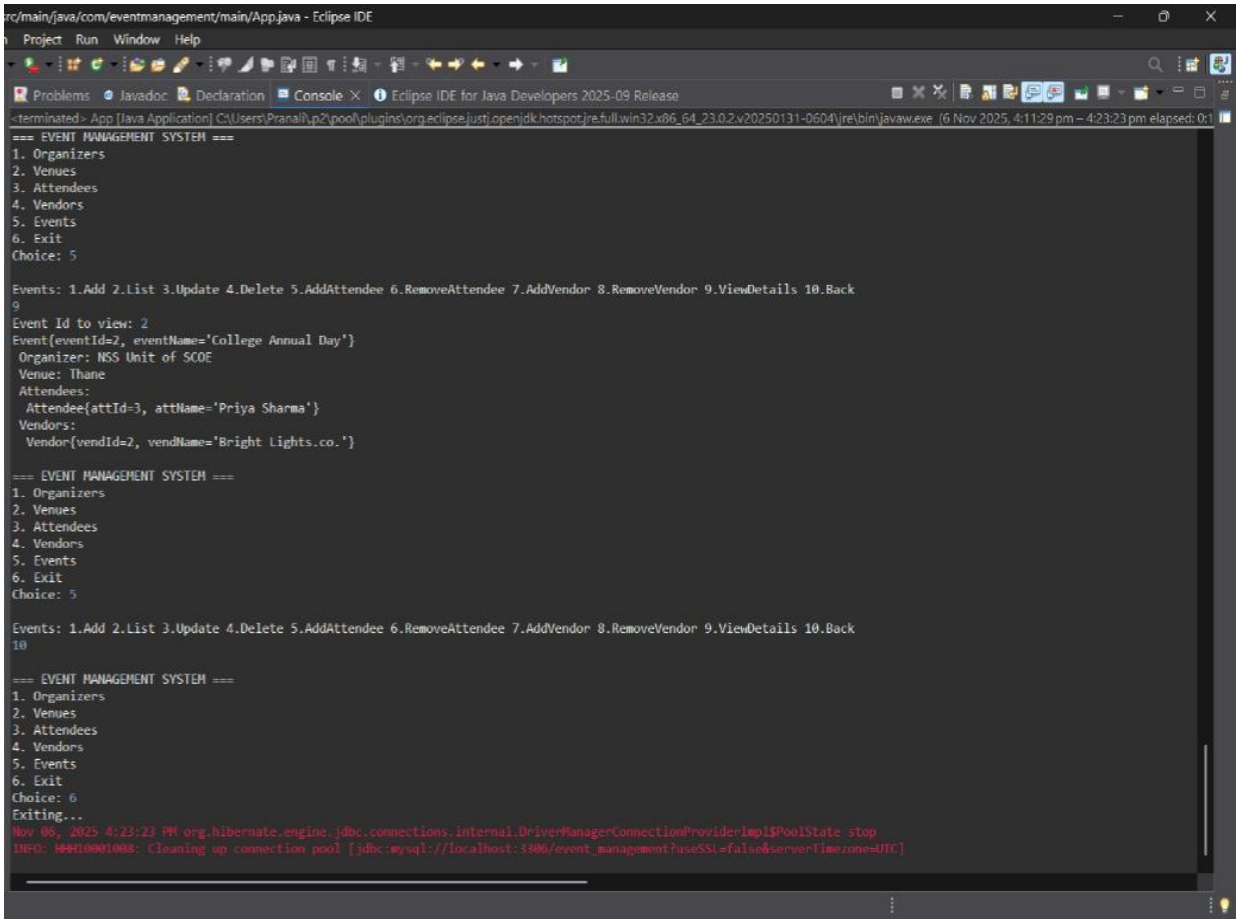

Fig. 3.7. Exiting the session after completing the task

4. Implementation

The implementation phase involves converting the system design into a functional software application. The Event Management System was developed as a Java console-based application using Hibernate ORM for database management and MySQL as the relational database. This section describes the technologies, logic, and design used in the implementation.

1. Technology Stack

- Programming Language: Java (JDK 17)
- Framework: Hibernate ORM (for Object-Relational Mapping)
- Database: MySQL (Relational Database Management System)
- Build Tool: Apache Maven
- Testing Framework: Junit
- Logging: SLF4J (Simple Logging Façade for Java)
- Development Environment: IntelliJ IDEA / Eclipse

2. Backend Implementation

The backend handles all core logic and database operations using Hibernate. CRUD functionalities are implemented for each entity: Organizer, Venue, Attendee, Vendor, and Event.

Key Functionalities:

- Organizer Management:
  Add, list, update, and delete organizers who manage events.
- Venue Management:
  Store and update venue details such as name, capacity, and contact.
- Event Management:
  Create events by assigning organizers and venues, and schedule them with date and time.
- Attendee & Vendor Management:
  Add or remove attendees and vendors linked to specific events via many-to-many relationships (event_attendee, event_vendor).

3. Console Interface Implementation

Since this is a console-based system, a menu-driven interface allows the user to perform CRUD operations.

The user enters values, which are captured through Java's Scanner class and passed to DAO methods for execution.

4. Database Implementation

The database is implemented using MySQL with seven main tables:

- Organizer – Stores organizer details.
- Venue – Stores venue data including capacity and location.
- Attendee – Contains information about event participants.
- Vendor – Stores vendor details and services offered.
- Event – Stores all event details and links organizers and venues.
- Event_attendee – Junction table for event–attendee relationships.
- Event_vendor – Junction table for event–vendor relationships.

Relationships:

- One Organizer → Many Events
- One Venue → Many Events
- One Event ↔ Many Attendees (M : M)
- One Event ↔ Many Vendors (M : M)

5. Security Implementation

- Foreign Key Constraints: Maintain relational integrity between entities.
- Transaction Management: All CRUD operations are wrapped in Hibernate transactions to prevent partial commits.
- Exception Handling: Ensures the system gracefully handles invalid inputs or connection failures.
- Data Validation: Basic checks (e.g., null values, invalid IDs) are performed during input and database operations.

5. Results and Discussion

The development and implementation of the Event Management System produced several notable outcomes. This section highlights the system's functional effectiveness, performance, and areas for potential improvement based on testing and execution.

1. Results

- **Functional Results**
  - **Organizer Module:**
    Users can add, view, update, and delete organizer records. Organizer–Event linkage works correctly through foreign key associations.
  - Venue Module:
    Venues can be created and assigned to events. Capacity, location, and owner details are stored and retrieved accurately.
  - Event Module:
    Events can be created with assigned organizers and venues. Attendees and vendors can be added or removed dynamically from events.
  - Attendee Module:
    Allows complete CRUD operations and proper linking to multiple events through the event_attendee junction table.
  - Vendor Module:
    Supports adding, updating, and deleting vendor details and assigning them to multiple events via the event_vendor relationship.
  - Database Operations:
    All CRUD operations execute successfully using Hibernate, ensuring persistent and accurate data storage.

- **Non-Functional Results**
  - **Usability:**
    The console interface is simple, intuitive, and menu-driven, ensuring easy navigation for users.
  - Performance:
    Database queries and Hibernate transactions execute efficiently, maintaining quick response times for all operations.
  - Reliability:
    Proper exception handling ensures smooth system performance even

when invalid inputs are entered.

- Data Integrity:
  Relationships among entities (Organizer, Venue, Event, Attendee, Vendor) are enforced through Hibernate mappings and MySQL constraints.
- Scalability:
  The modular DAO structure allows easy addition of new features (e.g., ticketing, budgeting, or event analytics) in future versions.

2. Discussion

1. **Achievement of Objectives**
   - All major objectives—such as event creation, venue allocation, and management of attendees and vendors—were successfully achieved.
   - Relationships between different entities were correctly implemented, ensuring accurate linkage between events and related data.

2. **User Experience**
   - The menu-driven interface provides a smooth and structured experience for users.
   - Separate options for each module (Organizer, Venue, Vendor, Attendee, and Event) make operations straightforward and organized.

3. **Challenges Faced**
   - Configuring Hibernate mappings and resolving entity relationship issues required multiple iterations.
   - Maintaining referential integrity across junction tables (event_attendee, event_vendor) was initially challenging.
   - Handling date and time formats during event creation needed additional validation logic.

4. **Limitations**
   - The current version is console-based, lacking a graphical user interface.
   - There is no authentication system (e.g., login credentials) for organizers or users.
   - Reporting features are limited to text-based outputs rather than graphical dashboards.

5. **Future Enhancements**
   - Develop a web-based interface using Java Spring Boot or JSP for better interactivity.
   - Add login and authentication modules for organizers and admins.

- Include email notifications for event updates or venue changes.
- Generate automated event reports with summary charts.
- Implement search and filtering features for easier data access.

6. Conclusion

The development of the Event Management System has successfully demonstrated the design and implementation of a structured, efficient, and reliable platform for managing events.

The system enables users to:

- Create, manage, and organize events seamlessly.
- Register and maintain details of organizers, venues, attendees, and vendors.
- Establish relationships among these entities, ensuring smooth coordination.
- Perform all CRUD operations efficiently using a console-based interface

Key achievements include:

- Seamless integration between entities such as Event–Organizer, Event–Venue, Event–Attendee, and Event–Vendor, ensuring data consistency.
- Implementation of Hibernate ORM for simplified database interaction and reliable data persistence.
- A modular architecture that enhances scalability, allowing easy addition of new features and modules.
- Proper transaction management and exception handling, ensuring stability and reliability during operations.

Overall, the Event Management System provides a robust and organized solution for planning and executing events efficiently. It reduces manual effort, minimizes human errors, and streamlines event coordination — offering a solid foundation for future expansion into a web-based or enterprise-level event management platform.

7. Future scope

Although the Event Management System effectively meets its current objectives, several enhancements can be introduced to make the system more feature-rich, automated, and user-friendly in the future:

1. Web-Based Interface

   Develop a fully functional web or desktop GUI using JavaFX or Spring Boot to replace the console-based interface.

   Provide a responsive and visually appealing design for improved user experience.

2. Authentication & Role Management

   Implement secure login and authentication features for organizers and administrators.

   Introduce role-based access control (e.g., Admin, Organizer, Vendor) to enhance system security.

3. Notifications & Scheduling

   Integrate email or SMS notifications for event reminders, schedule changes, and vendor confirmations.

   Add an automated scheduling assistant to avoid venue conflicts and double bookings.

4. Reporting and Analytics

   Generate graphical reports and dashboards showing event attendance, vendor performance, and venue utilization.

   Implement export options for event summaries in PDF or Excel formats.

8. Bibliography and References

In preparing this project report on **Event Management System**, several books, research papers, and online resources were referred to. These materials provided valuable insights into system design, database normalization, software development methodologies, and event coordination processes.

Books

1. Rajaraman, V. – *Fundamentals of Computers*, Prentice Hall of India.
2. Abraham Silberschatz, Henry Korth, S. Sudarshan – *Database System Concepts*, McGraw Hill.
3. Roger S. Pressman – *Software Engineering: A Practitioner's Approach*, McGraw Hill.
4. Ian Sommerville – *Software Engineering*, Pearson Education.
5. Herbert Schildt – *Java: The Complete Reference*, McGraw Hill Education.

Research Papers / Journals

1. International Journal of Computer Applications (IJCA) – Studies on Event Management Systems and Information Systems Integration.
2. IEEE Xplore Digital Library – Articles related to Event Planning Automation and Database-driven Management Systems.
3. Springer Open Journal of Software Engineering – Publications on Object-Oriented Design and System Architecture.

Websites

1. https://hibernate.org/ - Official Hibernate ORM documentation.
2. https://www.mysql.com/ - MySQL official documentation.
3. https://maven.apache.org/ - Apache Maven project management tool documentation.
4. https://docs.oracle.com/javase/ - Official Java SE documentation.
5. https://www.baeldung.com/hibernate-tutorial - Tutorials and best practices for Hibernate in Java.