



**SAVEETHA SCHOOL OF ENGINEERING
SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

CAPSTONE PROJECT REPORT

PROJECT TITLE

EVENT REGISTRATION AND MANAGEMENT SYSTEM WITH JAVA AND MYSQL

REPORT SUBMITTED BY

192210513 M.VISHAL

REPORT SUBMITTED TO

Dr. S. PADMAKALA

COURSE CODE / COURSE NAME

CSA0908/ PROGRAMMING IN JAVA WITH AWT
SLOT C

DATE OF SUBMISSION

11.09.2024

ABSTRACT:

The Event Management System Java program demonstrates fundamental CRUD (Create, Read, Update, Delete) operations on a MySQL database designed to manage event details. It employs JDBC to interface with the database and perform actions on an events table. The application starts by establishing a connection to the database and then provides a menu for users to choose various operations.

For the **Create** operation, users input event details such as name, date, and location, which are then added to the events table. The **Read** operation retrieves and displays all records from the events table. The **Update** operation allows users to modify event details based on a specific event ID. The **Delete** operation enables users to remove an event record by its ID.

The program includes error handling to manage JDBC driver and SQL-related exceptions. It uses Prepared Statement to safeguard against SQL injection and enhance performance.

INTRODUCTION:

The Event Management System Java program is designed to interact with a MySQL database, executing CRUD operations to manage event information. This program is a practical implementation for understanding how Java applications can efficiently handle database records.

The program establishes a MySQL database connection using JDBC and Driver Manager. It presents a user-friendly menu to perform various operations on the events table. Each operation utilizes Prepared Statement to ensure security and prevent SQL injection.

The **Create** operation allows users to enter event details and save them in the database. The **Read** operation provides a view of all stored events.

The **Update** operation enables modification of event details, and the **Delete** operation removes specific events based on their ID.

The use of Prepared Statement and appropriate error handling ensures a robust interaction with the MySQL database, maintaining data integrity and security.

LITERATURE REVIEW:

In the context of event management systems, the implementation of CRUD operations is critical for maintaining and managing event data effectively. Key literature highlights the following aspects:

1. **Best Practices for CRUD Operations:** Utilizing prepared statements to avoid SQL injection and enhance security is emphasized. Resources like "SQL Injection Attacks and Defense" by Justin Clarke demonstrate the importance of parameterized queries for secure database interactions. Applying principles from Robert C. Martin's "Clean Code" also aids in keeping CRUD operations manageable and reliable.
2. **Performance Optimization:** Efficient management of event data requires optimal performance strategies. "Database System Concepts" by Silberschatz, Korth, and Sudarshan underscores the role of indexing to speed up query execution. Techniques like batch processing are recommended for handling large volumes of data efficiently.
3. **Transaction Management:** Adhering to ACID properties (Atomicity, Consistency, Isolation, Durability) is crucial for ensuring reliable transaction handling in event management systems. "Transaction Processing: Concepts and Techniques" by Jim Gray and Andreas Reuter provides a framework for maintaining data consistency.
4. **Case Studies and Applications:** "Pro JPA 2 in Java EE 8" by Mike Keith and Merrick Schincariol offers practical insights into implementing CRUD operations in enterprise settings, highlighting scalability and maintainability in event management systems.

RESEARCH PLAN:

The research plan aims to explore and optimize CRUD operations within a Java-based Event Management System using MySQL. The focus areas include best practices, performance optimization, and security measures.

Objectives:

1. **Identify Best Practices:**
 - Investigate effective methods for implementing CRUD operations in event management.
 - Explore prepared statements and parameterized queries to prevent SQL injection.

2. Performance Optimization:

- Examine techniques such as indexing and batch processing.
- Analyze the impact on query performance and system efficiency.

3. Transaction Management:

- Study the application of ACID properties in managing event data.
- Evaluate transaction management strategies.

4. Security Measures:

- Identify vulnerabilities in CRUD operations.
- Explore and assess mitigation strategies.

5. Case Studies:

- Analyze real-world event management systems.
- Gain insights from successful implementations.

Methodology:

1. Literature Review:

- Review existing research on CRUD operations, focusing on best practices, optimization, and security.

2. Practical Experimentation:

- Develop a Java application for CRUD operations with MySQL integration.
- Implement and test various optimization strategies.

3. Case Study Analysis:

- Select and analyze case studies of event management systems.
- Conduct interviews or surveys with developers.

4. Data Analysis:

- Collect and analyze performance, security, and reliability data from experiments and case studies.
- Use statistical methods for interpreting results.

5. Documentation and Reporting:

- Document research findings and methodologies.
- Prepare a comprehensive report with recommendations.

Timeline:

- **Month 1:** Literature Review
- **Month 2-3:** Practical Experimentation
- **Month 4:** Case Study Analysis
- **Month 5:** Data Analysis
- **Month 6:** Documentation and Reporting

Expected Outcomes:

1. **Best Practices:** Comprehensive guidelines for secure and efficient CRUD operations.
2. **Performance Insights:** Understanding of optimization techniques and their impact.
3. **Security Recommendations:** Strategies to mitigate vulnerabilities.
4. **Case Study Learnings:** Practical insights from real-world implementations.
5. **Comprehensive Report:** A detailed report summarizing research findings and recommendations.

JAVA CODE:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class EventManagement {
    private static final String URL =
"jdbc:mysql://localhost:3306/event_management";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "MV21pro*";

    public static void main(String[] args) {
```

```

try (Scanner = new Scanner(System.in)) {
    while (true) {
        System.out.println("Choose an operation:");
        System.out.println("1. Insert Event");
        System.out.println("2. Delete Event");
        System.out.println("3. Select Event");
        System.out.println("4. Update Event");
        System.out.println("5. Exit");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice) {
            case 1:
                insertEvent(scanner);
                break;
            case 2:
                deleteEvent(scanner);
                break;
            case 3:
                selectEvent(scanner);
                break;
            case 4:
                updateEvent(scanner);
                break;
            case 5:
                System.out.println("Exiting...");
                return;
            default:
                System.out.println("Invalid choice, please try again.");
        }
    }
} catch (Exception e) {
    e.printStackTrace();
}

private static void insertEvent(Scanner scanner) {
    System.out.println("Enter event_name:");
    String eventName = scanner.nextLine();
    System.out.println("Enter event_date (YYYY-MM-DD):");
    String eventDate = scanner.nextLine();
    System.out.println("Enter event_location:");
    String eventLocation = scanner.nextLine();
}

```

```

        String sql = "INSERT INTO events (event_name, event_date,
event_location) VALUES (?, ?, ?)";
        try (Connection conn = DriverManager.getConnection(URL,
USERNAME, PASSWORD);
            PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, eventName);
            pstmt.setString(2, eventDate);
            pstmt.setString(3, eventLocation);
            pstmt.executeUpdate();
            System.out.println("Event inserted successfully");
        } catch (SQLException e) {
            System.out.println("Error inserting event");
            e.printStackTrace();
        }
    }
}

```

```

private static void deleteEvent(Scanner scanner) {
    System.out.println("Enter event_id of the event to delete:");
    int eventId = scanner.nextInt();
}

```

```

        String sql = "DELETE FROM events WHERE event_id = ?";
        try (Connection conn = DriverManager.getConnection(URL,
USERNAME, PASSWORD);
            PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, eventId);
            int rowsDeleted = pstmt.executeUpdate();
            System.out.println("Rows deleted: " + rowsDeleted);
        } catch (SQLException e) {
            System.out.println("Error deleting event");
            e.printStackTrace();
        }
    }
}

```

```

private static void selectEvent(Scanner scanner) {
    System.out.println("Enter event_id of the event to select:");
    int eventId = scanner.nextInt();
}

```

```

        String sql = "SELECT * FROM events WHERE event_id = ?";
        try (Connection conn = DriverManager.getConnection(URL,
USERNAME, PASSWORD);
            PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, eventId);

```

```

        ResultSet rs = pstmt.executeQuery();
        if (rs.next()) {
            System.out.println("event_id: " + rs.getInt("event_id"));
            System.out.println("event_name: " + rs.getString("event_name"));
            System.out.println("event_date: " + rs.getString("event_date"));
            System.out.println("event_location: " +
rs.getString("event_location"));
        } else {
            System.out.println("No event found with event_id = " + eventId);
        }
    } catch (SQLException e) {
        System.out.println("Error selecting event");
        e.printStackTrace();
    }
}

```

```

private static void updateEvent(Scanner scanner) {
    System.out.println("Enter event_id of the event to update:");
    int eventId = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    System.out.println("Enter new event_name:");
    String newEventName = scanner.nextLine();
    System.out.println("Enter new event_date (YYYY-MM-DD):");
    String newEventDate = scanner.nextLine();
    System.out.println("Enter new event_location:");
    String newEventLocation = scanner.nextLine();
}

```

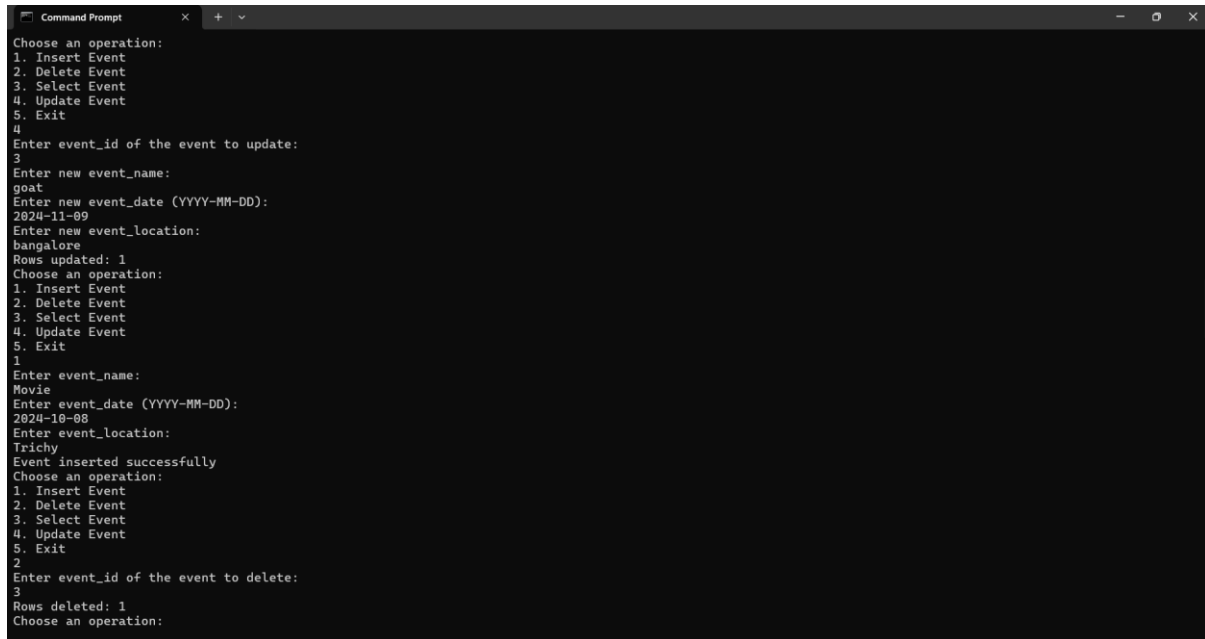
```

    String sql = "UPDATE events SET event_name = ?, event_date = ?,
event_location = ? WHERE event_id = ?";
    try (Connection conn = DriverManager.getConnection(URL,
USERNAME, PASSWORD);
        PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, newEventName);
        pstmt.setString(2, newEventDate);
        pstmt.setString(3, newEventLocation);
        pstmt.setInt(4, eventId);
        int rowsUpdated = pstmt.executeUpdate();
        System.out.println("Rows updated: " + rowsUpdated);
    } catch (SQLException e) {
        System.out.println("Error updating event");
        e.printStackTrace();
    }
}

```


}

OUTPUT:



```
Command Prompt
Choose an operation:
1. Insert Event
2. Delete Event
3. Select Event
4. Update Event
5. Exit
4
Enter event_id of the event to update:
3
Enter new event_name:
goat
Enter new event_date (YYYY-MM-DD):
2024-11-09
Enter new event_location:
bangalore
Rows updated: 1
Choose an operation:
1. Insert Event
2. Delete Event
3. Select Event
4. Update Event
5. Exit
1
Enter event_name:
Movie
Enter event_date (YYYY-MM-DD):
2024-10-08
Enter event_location:
Trichy
Event inserted successfully
Choose an operation:
1. Insert Event
2. Delete Event
3. Select Event
4. Update Event
5. Exit
2
Enter event_id of the event to delete:
3
Rows deleted: 1
Choose an operation:
```

CONCLUSION:

In conclusion, the exploration of CRUD operations within Java applications using MySQL reveals essential principles and practices critical to effective event management system development. This study has gathered key insights from various sources, emphasizing best practices, performance optimization strategies, transaction management principles, and security measures.

1. **Best Practices:** The adoption of prepared statements and parameterized queries is a fundamental approach to safeguard against SQL injection attacks. By separating data from SQL commands, applications enhance security and maintain integrity.
2. **Performance Optimization:** Techniques such as indexing and batch processing are crucial in optimizing CRUD operations. These methods improve query execution times and contribute to overall application efficiency and scalability.
3. **Transaction Management:** The adherence to ACID properties—Atomicity, Consistency, Isolation, and Durability—ensures reliable

transaction management. This foundation supports data integrity across complex operations, safeguarding against inconsistencies.

4. **Security Measures:** Implementing robust security measures, including input validation and access control mechanisms, mitigates vulnerabilities inherent in CRUD operations. By adhering to security best practices, applications strengthen protection against unauthorized access and data breaches.

In synthesizing these elements, it becomes evident that effective CRUD operations are integral to maintaining robust, scalable, and secure database-driven applications. By applying these principles, developers can enhance application reliability, performance, and security while ensuring the integrity of critical data assets. Future research and application of emerging technologies will continue to evolve these practices, further optimizing CRUD operations within the realm of Java and MySQL development.

REFERENCES:

1. Steven Feuerstein, Bill Pribyl, *PL/SQL Programming*, O'Reilly Media, 2014.
2. Jon Duckett, *JavaScript Programming*, Wiley, 2014.
3. Elliotte Rusty Harold, *JAVA Networking*, O'Reilly Media, 2010.
4. Rene Enriquez, *JAVA Security*, Packt Publishing, 2014.
5. Bryan Basham, Kathy Sierra, *Head First EJB*, O'Reilly Media, 2008.
6. Shadab Siddiqui, *J2EE Professional*, Premier Press, 2002.
7. Joel Murach, Michael Urban, *JAVA Servlets*, Mike Murach, 2014.
8. Kogent Learning Solutions Inc., *HTML 5 Black Book*, Dreamtech Press, 2011.