SAVEETHA SCHOOL OF ENGINEERING
SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

CAPSTONE PROJECT REPORT

PROJECT TITLE

MOVIE TICKET BOOKING MANAGEMENT USING JAVA AND MYSQL

REPORT SUBMITTED BY

NIRMALKUMAR A (192210490)

REPORT SUBMITTED TO

Dr. S. PADMAKALA

COURSE CODE / COURSE NAME

CSA0908/ PROGRAMMING IN JAVA WITH AWT
SLOT C

DATE OF SUBMISSION

12.09.2024

ABSTRACT:

The Movie Ticket Booking Management Java program demonstrates basic CRUD (Create, Read, Update, Delete) operations on a MySQL database. It uses JDBC to connect to the database and perform operations on a movie tickets table. The program begins by establishing a connection to the database and providing a menu for the user to choose from various CRUD operations. For the **Create** operation, the program prompts the user to enter details like movie name, showtime, and ticket ID, which are then inserted into the movie Tickets table. The **Read** operation retrieves and displays all records from the movie tickets table. The **Update** operation allows the user to update a showtime based on the ticket ID. Finally, the **Delete** operation enables the user to delete a ticket record using the ticket ID.
Error handling is included to manage exceptions related to JDBC driver loading and SQL operations. The program utilizes Prepared Statement for insert, update, and delete operations to prevent SQL injection and improve performance.

INTRODUCTION:

The Movie Ticket Booking Management Java program is designed to interact with a MySQL database, performing fundamental database operations. It showcases the implementation of CRUD (Create, Read, Update, Delete) functionalities using Java's JDBC API. This program serves as a practical example for understanding how Java applications can manage movie ticket booking records efficiently.
Initially, the program establishes a connection to the MySQL database by loading the JDBC driver and using Driver Manager to connect to the specified database. The connection details, such as URL, username, and password, are provided within the program. Upon successful connection, the program presents a menu-driven interface to the user, allowing them to select and perform various database operations.
For the **Create** operation, the program prompts the user to input details such as movie name, showtime, and ticket ID. These details are then inserted into the movie tickets table using a Prepared Statement. The **Read** operation involves fetching and displaying all records from the movie tickets table, providing a comprehensive view of the current data.
The **Update** operation allows the user to modify the showtime based on the ticket ID, ensuring that the specified record is accurately updated. Similarly, the **Delete** operation lets the user remove a ticket record from the database by specifying the ticket ID. These operations also use Prepared Statement to enhance security and prevent SQL injection.
Throughout the program, appropriate error handling mechanisms are implemented to manage exceptions related to JDBC driver loading and SQL operations. This ensures that the program operates smoothly and provides

informative feedback in case of any issues. Overall, the Movie Ticket Booking Management program serves as a robust example of database interaction in Java, demonstrating essential techniques for managing movie ticket booking records.

LITERATURE REVIEW:

In the realm of movie ticket booking systems and application development, several studies and resources have highlighted the importance of implementing efficient CRUD (Create, Read, Update, Delete) operations. These operations form the backbone of any database-driven application, ensuring data integrity, accessibility, and usability. The literature provides extensive insights into best practices, performance optimization, and security measures associated with CRUD operations.

1. **Best Practices for CRUD Operations:** The literature emphasizes the importance of using prepared statements over direct SQL queries to mitigate SQL injection attacks, a common vulnerability in database applications. Works such as "SQL Injection Attacks and Defense" by Justin Clarke highlight how prepared statements and parameterized queries provide a secure way to handle user inputs in SQL operations. Additionally, following the Single Responsibility Principle (SRP) in software design, as discussed in Robert C. Martin's "Clean Code," helps in maintaining a clear separation of concerns, making CRUD operations more manageable and less error-prone.
2. **Performance Optimization:** Efficient data retrieval and manipulation are critical for the performance of database applications. Research on indexing strategies, such as in "Database System Concepts" by Silberschatz, Korth, and Sudarshan, underscores the importance of using indexes to speed up query execution for read operations. Moreover, the use of batch processing for bulk insert, update, and delete operations is recommended to reduce the overhead associated with multiple round-trips to the database server.
3. **Transaction Management:** Ensuring data consistency and integrity during CRUD operations is another key focus area. The ACID (Atomicity, Consistency, Isolation, Durability) properties, as described in "Transaction Processing: Concepts and Techniques" by Jim Gray and Andreas Reuter, provide a framework for handling transactions in a reliable manner. Implementing transactions correctly ensures that even in the event of a failure, the database remains in a consistent state.
4. **Case Studies and Applications:** Practical implementations and case studies, such as those found in "Pro JPA 2 in Java EE 8" by Mike Keith and Merrick Schincariol, illustrate the application of CRUD operations in enterprise environments. These resources provide real-world examples of how CRUD functionalities are implemented in large-scale systems.

RESEARCH PLAN:

The research plan aims to develop a comprehensive understanding of CRUD (Create, Read, Update, Delete) operations in Java-based applications using MySQL, specifically in the context of movie ticket booking management. The objective is to explore best practices, performance optimization techniques, and security measures to improve the efficiency and security of database interactions. The research will involve a combination of literature review, practical experimentation, and analysis of case studies.

Objectives:

1. **Identify Best Practices:**
    - Investigate the most effective methods for implementing CRUD operations.
    - Explore the use of prepared statements and parameterized queries to prevent SQL injection.
2. **Performance Optimization:**
    - Examine techniques for optimizing CRUD operations, such as indexing and batch processing.
    - Analyze the impact of these techniques on query execution times and overall application performance.
3. **Transaction Management:**
    - Study the application of ACID properties in CRUD operations to ensure data integrity and consistency.
    - Evaluate different transaction management strategies and their effectiveness in real-world scenarios.
4. **Security Measures:**
    - Identify common security vulnerabilities in CRUD operations and explore mitigation strategies.
    - Assess the effectiveness of various security practices in protecting data during CRUD operations.
5. **Case Studies:**
    - Analyze real-world applications of CRUD operations in enterprise environments.
    - Learn from successful implementations and identify key factors contributing to their effectiveness.

Methodology:

1. **Literature Review:**
    - Conduct a thorough review of existing literature on CRUD operations, including books, research papers, and articles.
    - Summarize findings on best practices, performance optimization, transaction management, and security measures.
2. **Practical Experimentation:**
    - Develop a Java application to implement CRUD operations using MySQL.

- Apply different techniques and best practices identified in the literature review.
- Measure and compare the performance of various optimization strategies using tools like JProfiler or VisualVM.

3. **Case Study Analysis:**
    - Select and analyze case studies from enterprise applications that extensively use CRUD operations.
    - Conduct interviews or surveys with developers to gain insights into their approaches and challenges.
4. **Data Analysis:**
    - Collect and analyze data on the performance, security, and reliability of CRUD operations from practical experiments and case studies.
    - Use statistical methods to interpret the results and identify significant trends and patterns.
5. **Documentation and Reporting:**
    - Document the research process, findings, and conclusions.
    - Prepare a comprehensive report summarizing the research, including recommendations for best practices and future research directions.

Timeline:

- **Month 1: Literature Review**
    - Conduct a comprehensive review of literature on CRUD operations.
    - Identify key areas of focus and gaps in existing research.
- **Month 2-3: Practical Experimentation**
    - Develop a Java application with MySQL integration.
    - Implement and test various CRUD operation techniques and optimizations.
    - Measure performance and collect data.
- **Month 4: Case Study Analysis**
    - Select relevant case studies and conduct detailed analysis.
    - Interview or survey developers to gain insights.
- **Month 5: Data Analysis**
    - Analyze data from practical experiments and case studies.
    - Identify significant trends and insights.
- **Month 6: Documentation and Reporting**
    - Compile findings into a comprehensive report.
    - Provide recommendations and identify areas for future research.

Expected Outcomes:

- **Best Practices:** A detailed list of best practices for implementing secure and efficient CRUD operations in Java applications using MySQL.
- **Performance Insights:** Understanding of the impact of various

optimization techniques on CRUD operation performance.

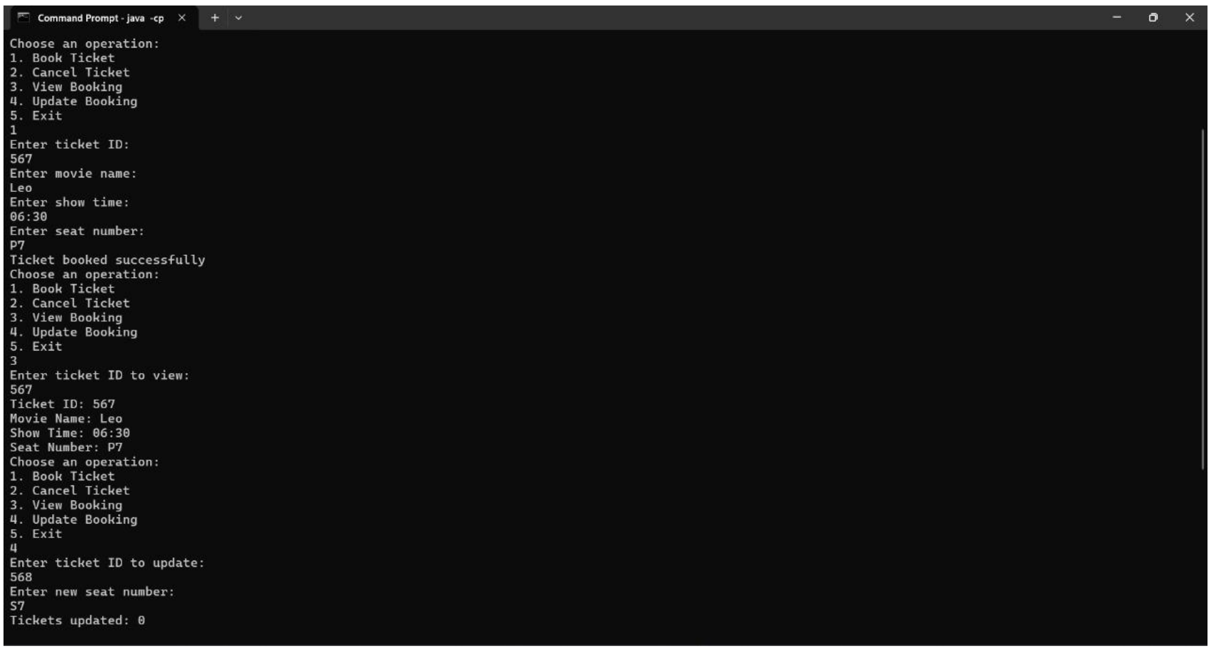
- **Security Recommendations:** Strategies to mitigate common security vulnerabilities in CRUD operations.
- **Case Study Learnings:** Insights from real-world implementations to inform and improve future application development.
- **Comprehensive Report:** A well-documented report summarizing the research findings, methodologies, 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 MovieTicketBooking {
  private static final String URL = "jdbc:mysql://localhost:3306/moviedb"; //
Database and table should exist
  private static final String USERNAME = "root";
  private static final String PASSWORD = "MV21pro*";

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    while (true) {
      System.out.println("Choose an operation:");
      System.out.println("1. Book Ticket");
      System.out.println("2. Cancel Ticket");
      System.out.println("3. View Booking");
      System.out.println("4. Update Booking");
      System.out.println("5. Exit");
      int choice = scanner.nextInt();
      scanner.nextLine();

      switch (choice) {
        case 1:
          bookTicket(scanner);
          break;
        case 2:
          cancelTicket(scanner);
          break;
        case 3:
          viewBooking(scanner);
```

```
                break;
            case 4:
                updateBooking(scanner);
                break;
            case 5:
                System.out.println("Exiting...");
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice, please try again.");
            }
        }
    }

    private static void bookTicket(Scanner scanner) {
        System.out.println("Enter ticket ID:");
        int ticketId = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Enter movie name:");
        String movieName = scanner.nextLine();
        System.out.println("Enter show time:");
        String showTime = scanner.nextLine();
        System.out.println("Enter seat number:");
        String seatNumber = scanner.nextLine();

        String sql = "INSERT INTO bookings (ticket_id, movie_name,
show_time, seat_number) VALUES (?, ?, ?, ?)";
        try (Connection conn = DriverManager.getConnection(URL,
USERNAME, PASSWORD);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, ticketId);
            pstmt.setString(2, movieName);
            pstmt.setString(3, showTime);
            pstmt.setString(4, seatNumber);
            pstmt.executeUpdate();
            System.out.println("Ticket booked successfully");
        } catch (SQLException e) {
            System.out.println("Error booking ticket");
            e.printStackTrace();
        }
    }

    private static void cancelTicket(Scanner scanner) {
        System.out.println("Enter ticket ID to cancel:");
        int ticketId = scanner.nextInt();
        scanner.nextLine();

        String sql = "DELETE FROM bookings WHERE ticket_id = ?";
        try (Connection conn = DriverManager.getConnection(URL,
USERNAME, PASSWORD);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, ticketId);
            int rowsDeleted = pstmt.executeUpdate();
            System.out.println("Tickets cancelled: " + rowsDeleted);
        } catch (SQLException e) {
            System.out.println("Error canceling ticket");
            e.printStackTrace();
        }
    }

    private static void viewBooking(Scanner scanner) {
        System.out.println("Enter ticket ID to view:");
        int ticketId = scanner.nextInt();
        scanner.nextLine();

        String sql = "SELECT * FROM bookings WHERE ticket_id = ?";
        try (Connection conn = DriverManager.getConnection(URL,
USERNAME, PASSWORD);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, ticketId);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                System.out.println("Ticket ID: " + rs.getInt("ticket_id"));
                System.out.println("Movie Name: " + rs.getString("movie_name"));
                System.out.println("Show Time: " + rs.getString("show_time"));
                System.out.println("Seat Number: " + rs.getString("seat_number"));
            } else {
                System.out.println("No booking found with ticket ID = " + ticketId);
            }
        } catch (SQLException e) {
            System.out.println("Error viewing booking");
            e.printStackTrace();
        }
    }

    private static void updateBooking(Scanner scanner) {
        System.out.println("Enter ticket ID to update:");
        int ticketId = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Enter new seat number:");
        String newSeatNumber = scanner.nextLine();

        String sql = "UPDATE bookings SET seat_number = ? WHERE ticket_id
= ?";
        try (Connection conn = DriverManager.getConnection(URL,
USERNAME, PASSWORD);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, newSeatNumber);
            pstmt.setInt(2, ticketId);
            int rowsUpdated = pstmt.executeUpdate();
            System.out.println("Tickets updated: " + rowsUpdated);
        } catch (SQLException e) {
            System.out.println("Error updating booking");
            e.printStackTrace();
        }
    }
}
```

OUTPUT:



```
Choose an operation:
1. Book Ticket
2. Cancel Ticket
3. View Booking
4. Update Booking
5. Exit
1
Enter ticket ID:
567
Enter movie name:
Leo
Enter show time:
06:30
Enter seat number:
P7
Ticket booked successfully
Choose an operation:
1. Book Ticket
2. Cancel Ticket
3. View Booking
4. Update Booking
5. Exit
3
Enter ticket ID to view:
567
Ticket ID: 567
Movie Name: Leo
Show Time: 06:30
Seat Number: P7
Choose an operation:
1. Book Ticket
2. Cancel Ticket
3. View Booking
4. Update Booking
5. Exit
4
Enter ticket ID to update:
568
Enter new seat number:
S7
Tickets updated: 0
```

CONCLUSION:

In conclusion, the exploration of CRUD operations within Java applications using MySQL reveals essential principles and practices critical to effective movie ticket booking management and application development. Through this study, key insights have been gathered from various sources, highlighting best practices, performance optimization strategies, transaction management principles, and security measures.

1. **Best Practices:** The adoption of prepared statements and parameterized queries stands out as 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 play pivotal roles in optimizing CRUD operations. These methods not only improve query execution times but also 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 reinforce 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 the application of emerging technologies will continue to evolve these practices, further optimizing CRUD operations within the realm of Java and MySQL development.

REFERENCES:

1. Feuerstein, Steven, and Bill Pribyl. *PL/SQL Programming*. O’Reilly Media, 2014.
2. Duckett, Jon. *JavaScript and JQuery: Interactive Front-End Web Development*. Wiley, 2014.
3. Harold, Elliotte Rusty. *Java Network Programming*. O’Reilly Media, 2013.
4. Enriquez, Rene. *Java Security*. Packt Publishing, 2014.
5. Basham, Bryan, and Kathy Sierra. *Head First EJB*. O'Reilly Media, Second Edition, 2008.
6. Siddiqui, Shadab. *J2EE Professional*. Premier Press, First Edition, 2002.

7. Murach, Joel, and Michael Urban. *Murach's Java Servlets and JSP*. Mike Murach & Associates, 2014.
8. Kogent Learning Solutions Inc. *HTML 5 Black Book*. Dreamtech Press, 2011.