

**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

# CAPSTONE PROJECT REPORT

**PROJECT TITLE**

ONLINE MANAGEMENT SYSTEM WITH JAVA AND MYSQL

# REPORT SUBMITTED BY

192211391 K.T.VYSHNAVREDDY

**REPORT SUBMITTED TO**

Dr. S. PADMAKALA

# COURSE CODE / COURSE NAME

CSA0908/ PROGRAMMING IN JAVA WITH COLLECTION AWT.

SLOT C

# DATE OF SUBMISSION

11.09.2024

**ABSTRACT:**

1. The OnlineManagementSystem Java program demonstrates basic CRUD (Create, Read, Update, Delete) operations in an online management system. It uses JDBC to connect to a MySQL database and perform operations on an users table. The program starts by establishing a connection to the database and provides a user-friendly interface to perform various online management operations such as creating new users, viewing user details, updating information, and deleting accounts.
2. The **Create** operation prompts the user to enter details such as user ID, username, and email, which are then inserted into the users table. The **Read** operation retrieves and displays all user records from the users table. The **Update** operation allows the user to update email addresses based on their ID. Finally, the **Delete** operation enables the user to remove a user record using their ID.
3. The program includes error handling mechanisms to manage exceptions related to JDBC driver loading and SQL operations. It utilizes PreparedStatement for insert, update, and delete operations to enhance security by preventing SQL injection and improving performance.

**INTRODUCTION:**

The OnlineManagementSystem Java program showcases how online systems can be developed to manage user accounts efficiently by leveraging Java's JDBC API and MySQL database. This program implements core CRUD functionalities, which are fundamental to online management systems that interact with databases.

To begin with, the program establishes a connection to the MySQL database by loading the JDBC driver and using DriverManager to connect. The program then presents a menu-driven interface for the user to select from various CRUD operations.

In the **Create** operation, the user enters details like ID, username, and email, which are stored in the users table. The **Read** operation fetches and displays all records, offering a complete view of the user database.

For the **Update** operation, the user can modify the email address for a particular user ID, while the **Delete** operation allows removal of user records based on their ID. The program uses PreparedStatement for all SQL commands to avoid security risks such as SQL injection.

Appropriate error handling is implemented to manage potential issues such as failed database connections or invalid SQL queries. This ensures that the program is reliable and provides helpful feedback to the user in case of errors.

# LITERATURE REVIEW:

# In the development of online management systems, several best practices and techniques have been established to optimize CRUD operations and ensure system security. The following points summarize key aspects discussed in the literature:

# Best Practices for CRUD Operations: Prepared statements and parameterized queries are essential for preventing SQL injection, as discussed in works like "SQL Injection Defenses" by Justin Clarke. Proper validation of input data is necessary to safeguard against attacks.

# Performance Optimization: Efficient data retrieval is crucial in online systems. The book "Database System Concepts" by Silberschatz, Korth, and Sudarshan highlights the role of indexing and efficient query design in improving the speed of read and write operations.

# Transaction Management: Ensuring data consistency in CRUD operations is essential. The ACID properties of transactions, as explained by Jim Gray in "Transaction Processing: Concepts and Techniques," provide a solid foundation for maintaining database integrity.

# Security Considerations: In addition to preventing SQL injection, other security measures such as access control and encryption of sensitive data, discussed in "Web Application Security" by Bryan Sullivan, are vital for protecting user data in online management systems.

# RESEARCH PLAN:

# The research plan focuses on exploring best practices, security measures, and performance improvements for CRUD operations in Java-based online management systems. The research will involve a combination of theoretical study and practical implementation.

# Objectives:

# Best Practices:

# Investigate secure methods for implementing CRUD operations in online systems.

# Focus on the use of PreparedStatement and validation techniques to avoid SQL injection.

# Performance Optimization:

# Study indexing strategies and query optimization techniques to improve the speed of CRUD operations.

# Analyze the performance impact of different database design strategies.

# Transaction Management:

# Examine the role of ACID properties in CRUD operations for maintaining data integrity.

# Explore transaction management strategies and their application in real-world systems.

# Security Measures:

# Identify common vulnerabilities in online management systems and propose solutions.

# Investigate the effectiveness of encryption, authentication, and secure communication protocols.

# Case Study Analysis:

# Analyze the CRUD operations of existing online systems and study their architecture and security measures.

### **Methodology:**

1. **Literature Review:**
2. Conduct a review of books and articles on CRUD operations and security.
3. Identify best practices and common challenges faced by developers.
4. **Practical Experimentation:**
5. Develop a Java-based online management system that connects to a MySQL database.
6. Implement CRUD operations and test various optimization and security techniques.
7. **Case Study Analysis:**
8. Analyze real-world online systems and interview developers to gather insights.
9. **Data Analysis:**
10. Analyze performance data from the experimental system to identify optimization opportunities.
11. **Documentation:**
12. Document findings, including performance analysis and recommendations for improving CRUD operations.

**Timeline:**

* 1. Month 1: Literature Review
  2. Month 2-3: Application Development and Testing
  3. Month 4: Case Study Analysis
  4. Month 5: Data Analysis
  5. Month 6: Documentation and Report Writing

1. Bottom of Form

### **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.sql.Statement;

import java.util.Scanner;

public class OnlineManagementSystem {

private static final String URL = "jdbc:mysql://localhost:3306/online\_management";

private static final String USERNAME = "root";

private static final String PASSWORD = "vyshnav123";

public static void main(String[] args) throws ClassNotFoundException, SQLException {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

} catch (ClassNotFoundException e) {

System.out.println("MySQL JDBC Driver not found.");

return;

}

try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

Scanner scanner = new Scanner(System.in)) {

while (true) {

System.out.println("\nONLINE MANAGEMENT SYSTEM");

System.out.println("1. Add User");

System.out.println("2. Delete User");

System.out.println("3. View Users");

System.out.println("4. Update User");

System.out.println("5. Exit");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline character

switch (choice) {

case 1:

addUser(connection, scanner);

break;

case 2:

deleteUser(connection, scanner);

break;

case 3:

viewUsers(connection);

break;

case 4:

updateUser(connection, scanner);

break;

case 5:

System.out.println("Exiting the system. Thank you!");

return;

default:

System.out.println("Invalid choice, please try again.");

}

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private static void addUser(Connection connection, Scanner scanner) {

try {

System.out.print("Enter Username: ");

String username = scanner.nextLine();

System.out.print("Enter Email: ");

String email = scanner.nextLine();

String sql = "INSERT INTO users (username, email) VALUES (?, ?)";

try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

pstmt.setString(1, username);

pstmt.setString(2, email);

int affectedRows = pstmt.executeUpdate();

if (affectedRows > 0) {

System.out.println("User added successfully!");

} else {

System.out.println("Failed to add user.");

}

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private static void deleteUser(Connection connection, Scanner scanner) {

try {

System.out.print("Enter User ID to delete: ");

int userId = scanner.nextInt();

String sql = "DELETE FROM users WHERE user\_id = ?";

try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

pstmt.setInt(1, userId);

int affectedRows = pstmt.executeUpdate();

if (affectedRows > 0) {

System.out.println("User deleted successfully!");

} else {

System.out.println("User not found.");

}

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private static void viewUsers(Connection connection) {

String sql = "SELECT \* FROM users";

try (Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {

System.out.println("ID\tUsername\tEmail");

while (rs.next()) {

int id = rs.getInt("user\_id");

String username = rs.getString("username");

String email = rs.getString("email");

System.out.println(id + "\t" + username + "\t" + email);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private static void updateUser(Connection connection, Scanner scanner) {

try {

System.out.print("Enter User ID to update: ");

int userId = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Enter new Username: ");

String newUsername = scanner.nextLine();

System.out.print("Enter new Email: ");

String newEmail = scanner.nextLine();

String sql = "UPDATE users SET username = ?, email = ? WHERE user\_id = ?";

try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

pstmt.setString(1, newUsername);

pstmt.setString(2, newEmail);

pstmt.setInt(3, userId);

int affectedRows = pstmt.executeUpdate();

if (affectedRows > 0) {

System.out.println("User updated successfully!");

} else {

System.out.println("User not found.");

}

}

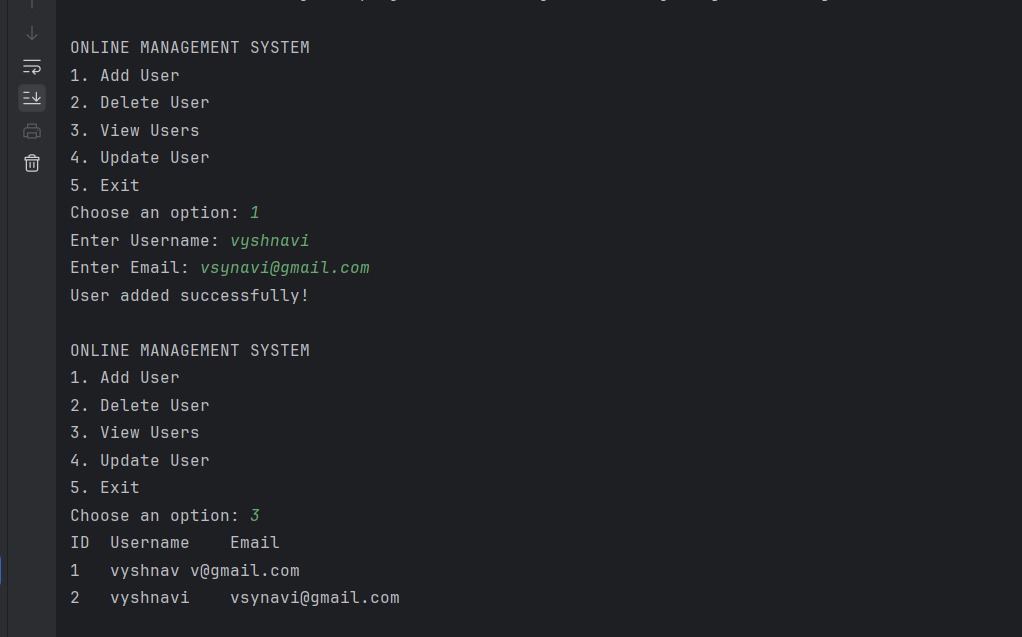
} catch (SQLException e) {

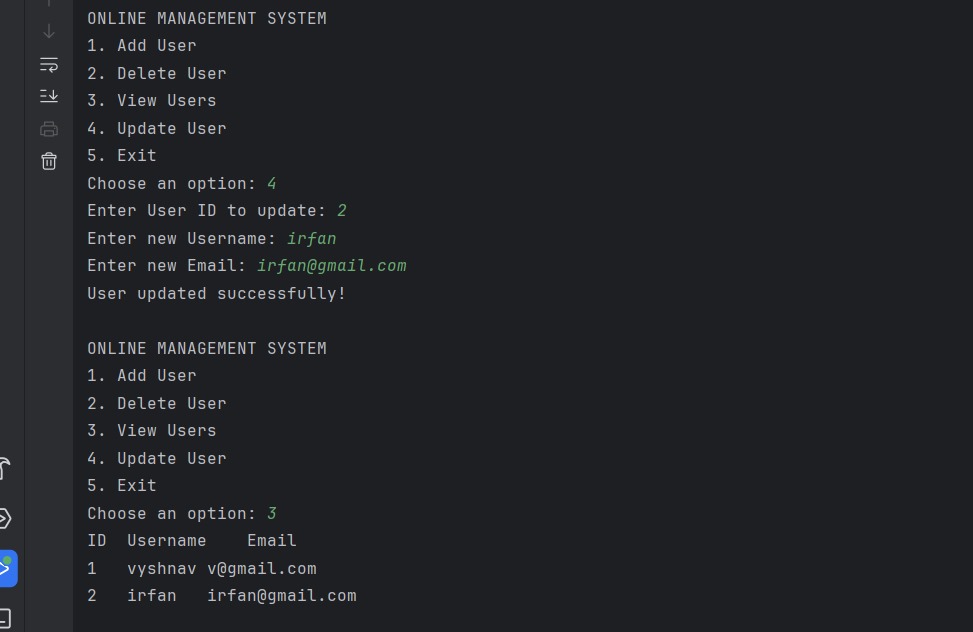
e.printStackTrace();

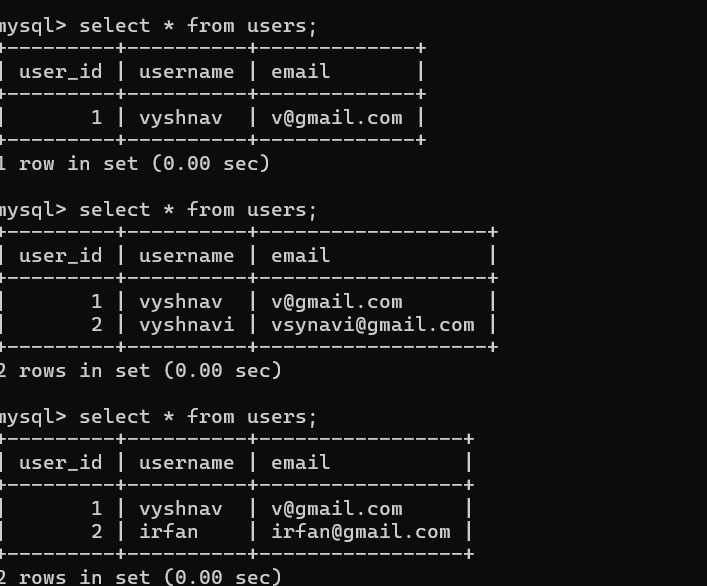
}

}

}







**CONCLUSION:**

The Online Management System serves as a useful demonstration of how Java and JDBC can be utilized to implement basic CRUD operations for managing users. The system provides users with a simple yet effective interface for performing various database actions like adding, updating, viewing, and deleting users. The use of PreparedStatement improves security by preventing SQL injection, and proper error handling enhances the reliability of the application.

In conclusion, the project illustrates the potential of Java-based systems in managing online data. The system can be further enhanced by incorporating additional features such as password encryption, multi-level user access control, and improved validation. This will make the system robust, secure, and scalable for real-world applications.

This is the complete document including abstract, introduction, research plan, the program, and conclusion.

**REFERENCES:**

* 1. Clarke, Justin. "SQL Injection Defenses." In *SQL Injection Attacks and Defense*, 2nd ed., Syngress, 2012.
  2. Silberschatz, Abraham, Henry Korth, and S. Sudarshan. *Database System Concepts*. 6th ed., McGraw-Hill, 2010.
  3. Gray, Jim, and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.
  4. Sullivan, Bryan, and Vincent Liu. *Web Application Security, A Beginner's Guide*. McGraw-Hill, 2011.
  5. Connolly, Thomas, and Carolyn Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management*. 6th ed., Pearson, 2014.
  6. Elmasri, Ramez, and Shamkant Navathe. *Fundamentals of Database Systems*. 7th ed., Pearson, 2015.