

EMPLOYEE MANAGEMENT SYSTEM

CS23333 – Object Oriented Programming using JAVA

Mini Project Report Submitted by

SUVATHI D - 231001225

SRI SANJANA B - 231001214

SHREENITHI A - 231001196

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE,

THANDALAM

(An Autonomous Institution)



2024

RAJALAKSHMI ENGINEERING COLLEGE

BONA FIDE CERTIFICATE

Certified that this project report titled “**EMPLOYEE MANAGEMENT SYSTEM**” is the bona fide work of “**SUVATHI D(231001225), SRI SANJANA B(231001214),SHREENITHI A(231001196)**” who Carried out the project work under my supervision.

Submitted for the Practical Examination held on_____

HEAD/IT

Dr.P.Valarmathie

Professor and Head,
Information Technology,
Rajalakshmi Engineering College
Thandalam, Chennai – 602 105

SUPERVISOR

Mr.K E Narayana,

Assistant Professor,
Information Technology
Rajalakshmi Engineering College
Thandalam, Chennai – 602 105

ABSTRACT

The main objective of this project is to build an Employee management system that will store and retrieve records of Employees. The System is intended to accept process, generate Employee details. The System was developed using basic technologies such as MYSQL database and Java (JDBC, SWING, AWT).The system is free of errors and very efficient and less time consuming due to care taken to develop it. All the phases of Development cycle are employed and it is worthwhile to state that the system is user friendly and strong

TABLE OF CONTENTS

1	INTRODUCTIONEMPLOYEE MANAGEMENT SYSTEM	6
	1.1 Purpose	
	1.2 Scope of Project	
	1.3 Software Requirement Specification	
2	SYSTEM FLOW DIAGRAMS	13
	2.1 Use Case Diagram	
	2.2 Entity-relationship Diagrams	
	2.3 Data Flow Diagram	
3	MODULE DESCRIPTION	15
	3.2 Add Employee	
	3.3 View Empl0yee	
	3.4 Delete Employee	
	3.5 Add Salary	
	3.6 Update Salary	
4	IMPLEMENTATION	16
	4.1 Design	
	4.2 Database Design	
	4.3Code	
5	CONCLUSION	32
6	REFERENCE	32

1 INTRODUCTION:

Employee Management System is a Software which is helpful for the Authorities. In the current system all the activities are done manually. It is very time consuming and costly. Our Employee Management System deals with the various activities related to the Employees.

By using this software, we can get Name and Salary details of the Employees in a single window page

1.1 PURPOSE:

The objective of *Employee management system* is to allow the administrator of any organization to edit and find out the personal details of a Employee and allows them to keep up to date their profile. Overall, it will make Employee information management an easier job for the administration and the Employee of any organization.

1.2 SCOPE OF THE PROJECT:

The proposed system will work with the administrator. The system works and fulfills all the functionalities as per the proposed system. It will provide reduced response time against the queries made by different users. This project is based on Java (JDBC) language with MYSQL database which manage the details of the Employee because it is a tedious job for any organization. Employee Information system will store all the details of the Employees including their background information.

1.3 Software requirement Specification:

Introduction:

The Employee management system can handle all the details about a Employee. The details include Employee personal details, Salary details etc., the Employee management system is an automated version of manual Employee management system.

Document Purpose:

This SRS Document contains the complete software requirements for the Employee Management System and describes the design decisions, architectural design and the detailed design needed to implement the system. It provides the visibility in the design and provides information needed for software support.

Product Scope:

Employee Management System is developing for general purpose and used to replace old paper work system. SMS is to build to efficiently provide Employee information to school/college administration. It provides a mechanism to edit the Employee information form which makes the system flexible.

Definitions, Acronyms and

Abbreviations: EMS - Online Employee Information
Management System SRS - Software Requirements
Specification

References and Acknowledgement:

- [1] <https://www.javatpoint.com/java-awt>
- [2] <https://www.javatpoint.com/java-swing>

Overall Description:

The Employee management system allows authorized members to access the records of academically registered Employees. It can be used in various educational institutes across the globe and simplifies working of institutes.

Product Perspective:

The proposed system shall be developed using client/server architecture and be compatible with Microsoft Windows Operating System. The front end of the system will be developed using Java AWT and SWING and backend will be developed using My SQL server.

Product Functionality:

- a) Add Employee: to store a new Employee.
- b) View Employee: to view and update an existing Employee
- c) Delete Employee: to delete an existing Employee.
- d) Add Salary: to store a particular Employee Salary.
- e) Update Salary: to view and update an existing Employee Salary.

User and Characteristics:

Qualification: At least matriculation and comfortable with English.

Experience: Should be well versed/informed about the registration process of the university.

Technical Experience: Elementary knowledge of computers.

Operating Environment:

Hardware requirements:

- Any Processor Over i3.
- Any version of Windows 8,10,11.
- Processor speed: 2.0 GHz
- RAM: 4GB
- Hard disk: 500GB

Software requirements:

- Database: MySQL
- Frontend: JAVA (SWING, AWT)

Technology: JAVA (JDBC) **Constraints:**

It will only be accessed by the administrator.

The delete operation is available only to the administrator.

To reduce the complexity of the system, there is no check on delete operation.

Hence, administrator should be very careful before deletion of any record and he/she will be responsible for data consistency.

Specific Requirements:

User Interface:

The EMS will have following user-friendly single page interface. a) Add

Employee: to store a new Employee.

b) View Employee: to view and update a existing Employee

c) Delete Employee: to delete a existing Employee.

d) Add Salary: to store a particular Employee Salary.

e) Update Salary: to view and update a existing Employee Salary

Hardware Interface:

a) Screen resolution of at least 640 x 480 or above.

b) **Any version of Windows 8,10,11.**

Software Interface:

a) MS-Windows Operating System

b) Java AWT and SWING for designing front-end

c) MS SQL Server for backend

d) PLATFORM : JAVA LANGUAGE

e) INTEGRATED DEVELOPMENT ENVIRONMENT(IDE): VSCODE

Functional Requirements:

Log in Module (LM):

User (admin) shall be able to load the Login Module. The LM shall support the user to log into the system. The login panel shall contain fields to contain a user name and a field for password. The password field shall be masked with symbols when the user types. It shall also contain a button labeled as Login. When the user clicks on Login button the username and password will be verified by database administrator and then only the user will able to use the system functions.

Registered Users Module (RUM):

After successful login, user shall be able to continue navigating through the Application and view Employee detailed information. After successful login, user (admin) shall be able to update and maintain their Employee profile, such as changing Salary and personal details.

Administrator Module (AM):

After successful login, system shall display administrative functions. Administrative functions shown shall be add and update. When administrator clicks on the add button, system shall display a section where administrator can add new Employee details, remove unused Employee details and many more. When administrator clicks on update button, system shall display a section where administrator can update Employee details which are currently stored in the database. When administrator adds, updates or delete an entry, the AM module will send the request to the Server Module which will do the necessary changes to the DB.

Server Module (SM):

SM shall be between the various modules and the DB. SM shall receive all requests and format the pages accordingly to be displayed. SM shall validate and execute all requests from the other modules

Non-functional Requirements:

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, etc.).

Performance:

The system and the server must be capable of handling the real-time error functionality occurs by the defined users. In addition, the system must be safety critical. All failures reported by the server side must be handled instantaneously to allow for user and system safety.

Reliability:

The system is safety critical. If it moves out of normal operation mode, the requirement to drop or down the server and fix it as soon as possible and open it again. This emergency behaviour shall not occur without reason.

Availability:

When in normal operating conditions, request by a user for system shall be handled within 1 second. Immediate feedback of the systems activities shall be communicated to the user. **Security:**

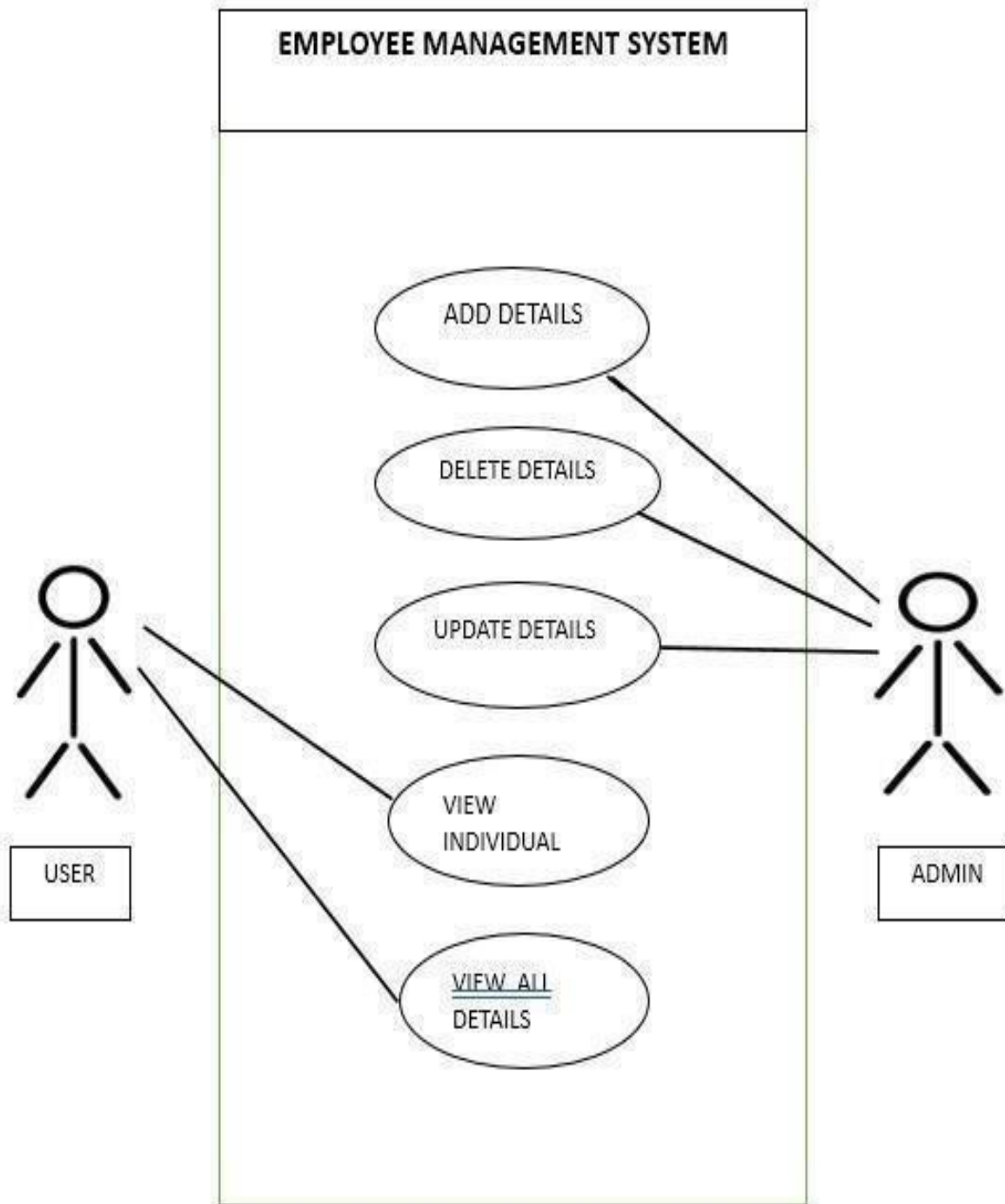
There shall be a strong security mechanism should be place in the server side of the system to keep unwanted users to hack or damage the system. However, all users of the system give and store the details of privacy related to personal information and many other.

Maintainability:

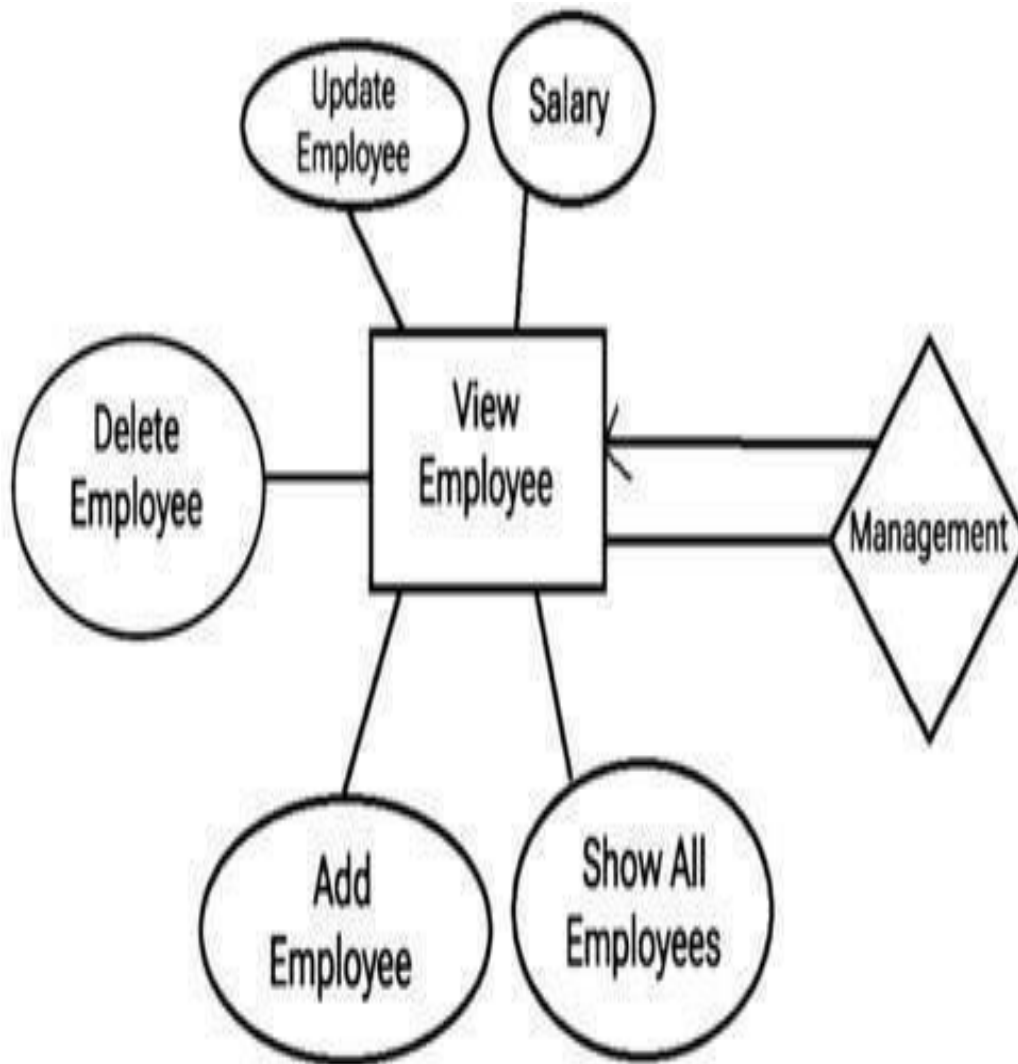
There shall be design documents describing maintenance of the software and database used to save the user details as well as the daily updated and modification done in system. There shall be an access on the control system by the admin to be maintained it properly at the front end as well as at back end.

SYSTEM FLOW DIAGRAMS:

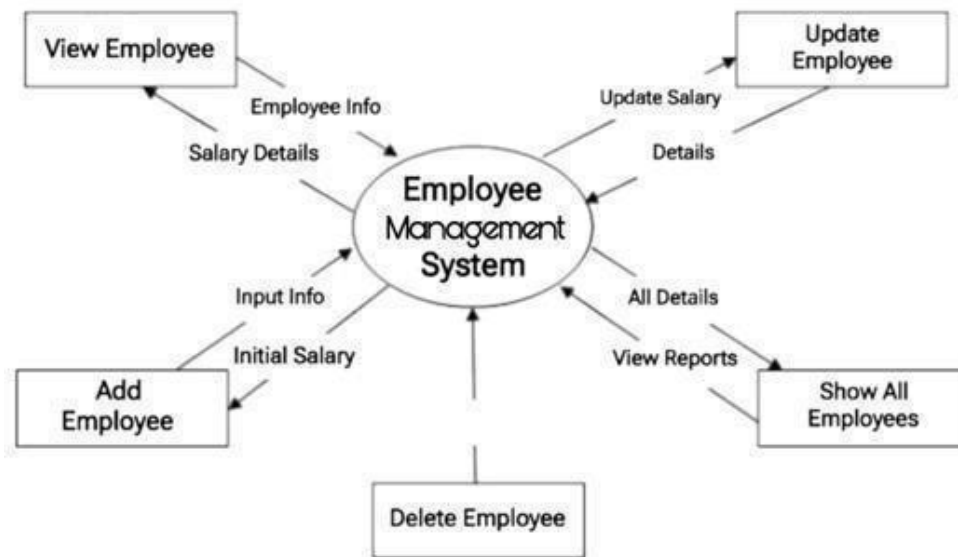
2.1 Use case Diagram:



2.2 ER Model Diagram:



2.3 Data Flow Diagram:



Employee id	Name	Salary
1	Mani	10000
2	Kamal	9999
3	Kailash	10000

3. MODULE DESCRIPTION:

The different types of modules present in this project are:

Details:

Employee id, Name, Salary

3.1 Add Employee: In this section admin can add Employee details

3.2 View Employee: In this section admin can view and update Employee details.

3.3 Delete Employee: In this section admin can delete Employees details.

3.4 Update Employee: In this section admin can update Employee Salary

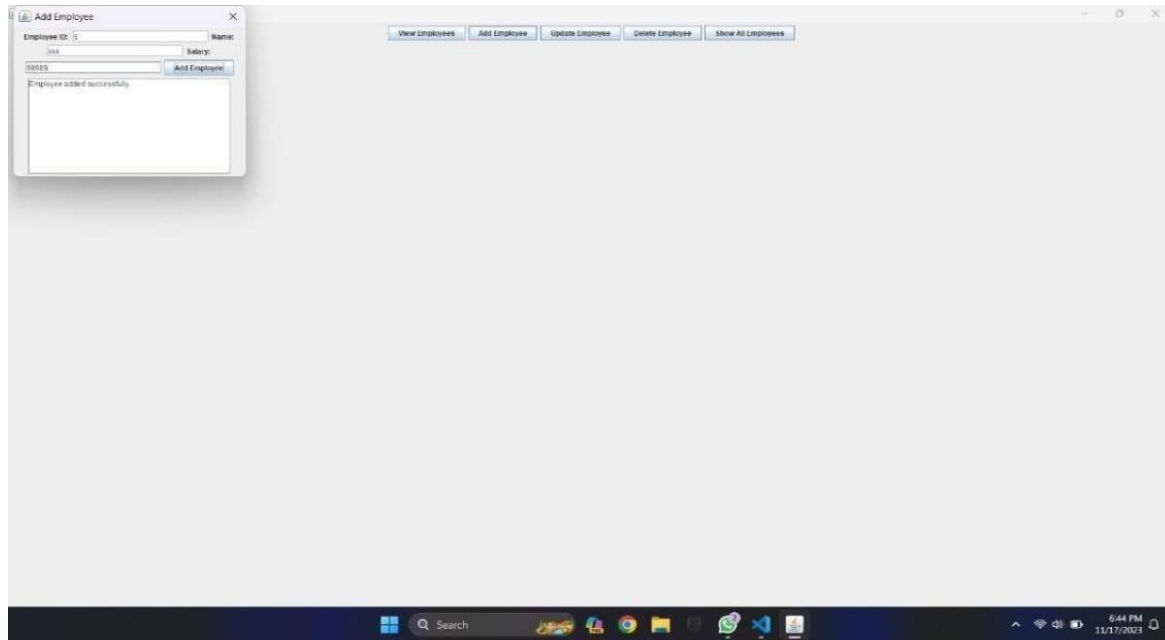
3.5 View Employee: In this section user can view details of individual Employee

3.6 Show all: In this section user view details of all the employees.

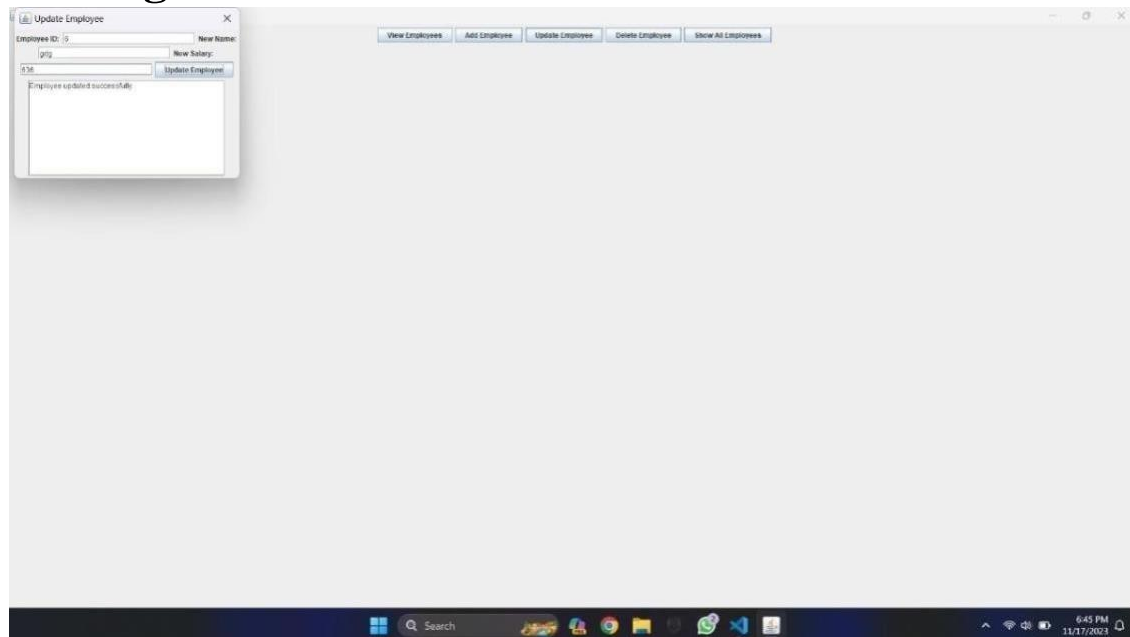
4 IMPLEMENTATION :

4.1 DESIGN :

SCREENS: Homepage



Adding Details



4.2 DATABASE DESIGN:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: ☐

	employee_id	name	salary
▶	1	m	22.00
	2	g	5.00
	3	kamal	999999.00
	4	raj	74747.00
	5	manu	5.00
•	NULL	NULL	NULL

Result Grid
Form Editor
Field Types

4.3 CODE:

```
import javax.swing.*;    import
java.awt.*;              import
java.awt.event.ActionEvent; import
java.awt.event.ActionListener; import
java.sql.Connection;      import
java.sql.DriverManager;    import
java.sql.PreparedStatement; import
java.sql.ResultSet;        import
java.sql.SQLException;

public class EmployeeManagementApp extends JFrame implements
ActionListener { private static final String

    JDBC_URL =
    "jdbc:mysql://localhost:3306/employee_db";

    private static final String USERNAME = "root";
    private static final String PASSWORD = "root";

    private JButton viewButton, addButton, updateButton, deleteButton,
allEmpButton;

    public EmployeeManagementApp() {
        setTitle("Employee Management");
        setSize(400, 200);
        setDefaultCloseOperation(EXIT_ON_CLOSE)
        ; setLayout(new FlowLayout());

        viewButton = new JButton("View Employees"); addButton
```

```

= new JButton("Add Employee"); updateButton = new
JButton("Update Employee"); deleteButton = new
JButton("Delete Employee"); allEmpButton = new JButton("Show
All Employees");

viewButton.addActionListener(this); addButton.addActionListener(this);
updateButton.addActionListener(this);
deleteButton.addActionListener(this);
allEmpButton.addActionListener(this); add(viewButton); add(addButton);
add(updateButton); add(deleteButton); add(allEmpButton);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == viewButton) {
        openViewEmployeesWindow();
    } else if (e.getSource() == addButton) { openAddEmployeeWindow();
    } else if (e.getSource() == updateButton) { openUpdateEmployeeWindow();
    } else if (e.getSource() == deleteButton) { openDeleteEmployeeWindow();
    }     else     if     (e.getSource()     ==     allEmpButton)     {
        openShowAllEmployeesWindow();
    }
}

private void openViewEmployeesWindow() {

```

```

JDialog viewDialog = new JDialog(this, "View Employees", true);
viewDialog.setSize(600, 400); viewDialog.setLayout(new
FlowLayout());

```

```

JComboBox<String> searchCriteriaComboBox = new
JComboBox<>(new String[]{"Employee ID", "Name", "Salary"});

```

```

JTextField searchText = new JTextField(20);

```

```

JButton searchButton = new JButton("Search"); JTextArea

```

```

resultTextArea = new JTextArea(10, 50);

```

```

resultTextArea.setEditable(false);

```

```

searchButton.addActionListener(new ActionListener() {

```

```

    @Override public void actionPerformed(ActionEvent
    e) {

```

```

        try (Connection connection =
        DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD)) {

```

```

            String criteria = (String)
            searchCriteriaComboBox.getSelectedItem();

```

```

            String searchTerm = searchText.getText();

```

```

            String query; if

```

```

(criteria.equals("Employee ID")) { query = "SELECT * FROM employees
WHERE employee_id =
?";

```

```

        } else if (criteria.equals("Name")) {

```

```

            query = "SELECT * FROM employees WHERE name = ?";

```

```

        } else if (criteria.equals("Salary")) { try

```

```

        {

```

```

        query = "SELECT * FROM employees WHERE salary = ?";
    } catch (NumberFormatException ex) {
        resultTextArea.setText("Invalid salary value. Please enter a
valid numeric value.");
        return;
    }
    } else { resultTextArea.setText("Invalid search criteria.");
    return; }

    try (PreparedStatement statement =
connection.prepareStatement(query)) {
        if (criteria.equals("Employee ID") || criteria.equals("Name")) {
            statement.setString(1, searchTerm);
        }

        try (ResultSet resultSet = statement.executeQuery())
        { resultTextArea.setText(""); // Clear the text area
        while (resultSet.next()) { int id =
resultSet.getInt("employee_id"); String name =
resultSet.getString("name"); double salary =
resultSet.getDouble("salary");

            resultTextArea.append("Employee ID: " + id + ", Name: " +
name + ", Salary: " + salary + "\n");
        }
    }
} catch (SQLException ex) { ex.printStackTrace();

```

```

        }
    } catch (SQLException ex) { ex.printStackTrace();
    }
}
});

```

```

viewDialog.add(searchCriteriaComboBox);
viewDialog.add(searchText);
viewDialog.add(searchButton);          viewDialog.add(new
JScrollPane(resultTextArea)); viewDialog.setVisible(true);
}

```

```

private void openAddEmployeeWindow() {
    JDialog addDialog = new JDialog(this, "Add Employee", true);
    addDialog.setSize(400, 300); addDialog.setLayout(new
FlowLayout());

    JTextField idField = new JTextField(20); // Add employee ID field
    JTextField nameField = new JTextField(20);
    JTextField salaryField = new JTextField(20);
    JButton addButton = new JButton("Add Employee");

    JTextArea resultTextArea = new JTextArea(10, 30);
    resultTextArea.setEditable(false);

    addButton.addActionListener(new ActionListener() {

```

```

@Override          public          void
    actionPerformed(ActionEvent e) {
        try (Connection connection =
DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD)) {
            int id = Integer.parseInt(idField.getText()); // Get employee ID
from the input field

            String name = nameField.getText();
            double salary = Double.parseDouble(salaryField.getText());

            String insertQuery = "INSERT INTO employees (employee_id,
name, salary) VALUES (?, ?, ?)";
            try (PreparedStatement insertStatement =
connection.prepareStatement(insertQuery)) {
                insertStatement.setInt(1, id); // Set the employee ID in the query
                insertStatement.setString(2, name);
                insertStatement.setDouble(3, salary); int rowsAffected =
                insertStatement.executeUpdate(); if (rowsAffected > 0) {
                    resultTextArea.setText("Employee added successfully.");
                } else { resultTextArea.setText("Failed to add the
employee."); }
            } catch (SQLException ex) { ex.printStackTrace();
        }
    } catch (SQLException ex) { ex.printStackTrace();
    }
}
});

```

```

addDialog.add(new JLabel("Employee ID: ")); // Label for employee ID
addDialog.add(idField); // Input field for employee ID
addDialog.add(new JLabel("Name: ")); addDialog.add(nameField);
addDialog.add(new JLabel("Salary: ")); addDialog.add(salaryField);
addDialog.add(addButton);
addDialog.add(new JScrollPane(resultTextArea));

addDialog.setVisible(true);
}

```

```

private void openUpdateEmployeeWindow() {
    JDialog updateDialog = new JDialog(this, "Update Employee", true);
    updateDialog.setSize(400, 300); updateDialog.setLayout(new
    FlowLayout());

```

```

    JTextField idField = new JTextField(20);
    JTextField nameField = new JTextField(20);
    JTextField salaryField = new JTextField(20);
    JButton updateButton = new JButton("Update Employee"); JTextArea
    resultTextArea = new JTextArea(10, 30);
    resultTextArea.setEditable(false);

```

```

    updateButton.addActionListener(new ActionListener() {
        @Override          public          void
        actionPerformed(ActionEvent e) { try
            (Connection connection =
DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD)) {

```

```

int id = Integer.parseInt(idField.getText()); String newName
= nameField.getText();
double newSalary = Double.parseDouble(salaryField.getText());

String updateQuery = "UPDATE employees SET name = ?, salary
= ? WHERE employee_id = ?";

try (PreparedStatement updateStatement =
connection.prepareStatement(updateQuery))
    { updateStatement.setString(1, newName);
      updateStatement.setDouble(2, newSalary);
      updateStatement.setInt(3, id);
      int rowsAffected = updateStatement.executeUpdate(); if
      (rowsAffected > 0) { resultTextArea.setText("Employee
      updated successfully.");
        } else { resultTextArea.setText("Failed to update the employee.");
        }
      } catch (SQLException ex) { ex.printStackTrace();
    }
  } catch (SQLException ex) { ex.printStackTrace();
}
});

```

```

updateDialog.add(new JLabel("Employee ID: "));
updateDialog.add(idField);    updateDialog.add(new
JLabel("New          Name:          "));
updateDialog.add(nameField);

```

```

        updateDialog.add(new JLabel("New Salary: "));
        updateDialog.add(salaryField);
        updateDialog.add(updateButton);
        updateDialog.add(new JScrollPane(resultTextArea));
        updateDialog.setVisible(true);
    }

```

```

private void openDeleteEmployeeWindow() {
    JDialog deleteDialog = new JDialog(this, "Delete Employee", true);
    deleteDialog.setSize(400, 150); deleteDialog.setLayout(new
    FlowLayout());

    JTextField idField = new JTextField(20);
    JButton deleteButton = new JButton("Delete Employee"); JTextArea
    resultTextArea = new JTextArea(4, 30);
    resultTextArea.setEditable(false);

    deleteButton.addActionListener(new ActionListener() {
        @Override public void
        actionPerformed(ActionEvent e) {
            try (Connection connection =
DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD)) {
                int id = Integer.parseInt(idField.getText());

                String deleteQuery = "DELETE FROM employees WHERE
employee_id = ?";

                try (PreparedStatement deleteStatement =

```

```

connection.prepareStatement(deleteQuery)) {
    deleteStatement.setInt(1, id); int rowsAffected =
    deleteStatement.executeUpdate(); if (rowsAffected >
    0) {
        resultTextArea.setText("Employee deleted successfully."); }
    else { resultTextArea.setText("Failed to delete the employee.");
    }
    } catch (SQLException ex) { ex.printStackTrace();
    }
    } catch (SQLException ex) { ex.printStackTrace();
    }
    }
});

```

```

deleteDialog.add(new JLabel("Employee ID: "));
deleteDialog.add(idField);
deleteDialog.add(deleteButton); deleteDialog.add(new
JScrollPane(resultTextArea));

```

```

deleteDialog.setVisible(true);
}

```

```

private void openShowAllEmployeesWindow() {

```

```

JDialog allEmpDialog = new JDialog(this, "All Employees", true);
allEmpDialog.setSize(600, 400); allEmpDialog.setLayout(new
FlowLayout());

JTextArea resultTextArea = new JTextArea(10, 50);
resultTextArea.setEditable(false);

try (Connection connection = DriverManager.getConnection(JDBC_URL,
USERNAME, PASSWORD)) {
    String query = "SELECT * FROM employees";

    try (PreparedStatement statement =
connection.prepareStatement(query)) {
        try (ResultSet resultSet = statement.executeQuery()) {
            resultTextArea.setText(""); // Clear the text area while
            (resultSet.next()) { int id =
                resultSet.getInt("employee_id"); String name
                = resultSet.getString("name"); double salary =
                resultSet.getDouble("salary");

                resultTextArea.append("Employee ID: " + id + ", Name: " +
name + ", Salary: " + salary + "\n");
            }
        }
    } catch (SQLException ex) { ex.printStackTrace();
}
} catch (SQLException ex) { ex.printStackTrace();
}

```

```

allEmpDialog.add(new JScrollPane(resultTextArea));

allEmpDialog.setVisible(true);
}

public static void main(String[] args)
{ SwingUtilities.invokeLater(() -> {
    EmployeeManagementApp app = new EmployeeManagementApp();
    app.setVisible(true);
});
}
}

```

TABLE CREATION:

```

import java.sql.Connection; import
java.sql.DriverManager; import
java.sql.SQLException; import
java.sql.Statement;

public class CreateTable {

public static void main(String[] args) {
    // Database connection parameters
    String JDBC_URL = "jdbc:mysql://localhost:3306/employee_db";
    String USERNAME = "root";

```

```

String PASSWORD = "root";

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

    Statement statement = connection.createStatement()) {

    // SQL query to create the "employees" table

    String createTableSQL = "CREATE TABLE employees (" +
        "employee_id INT AUTO_INCREMENT PRIMARY KEY," +
        "name VARCHAR(255) NOT NULL," +
        "salary DECIMAL(10, 2)" +
        //"CREATE SEQUENCE employee_id_seq START WITH 1
INCREMENT BY 1;" +
        ")";

    // Execute the SQL query to create the table
    statement.execute(createTableSQL);

    System.out.println("Table 'employees' created successfully.");

} catch (SQLException e) {

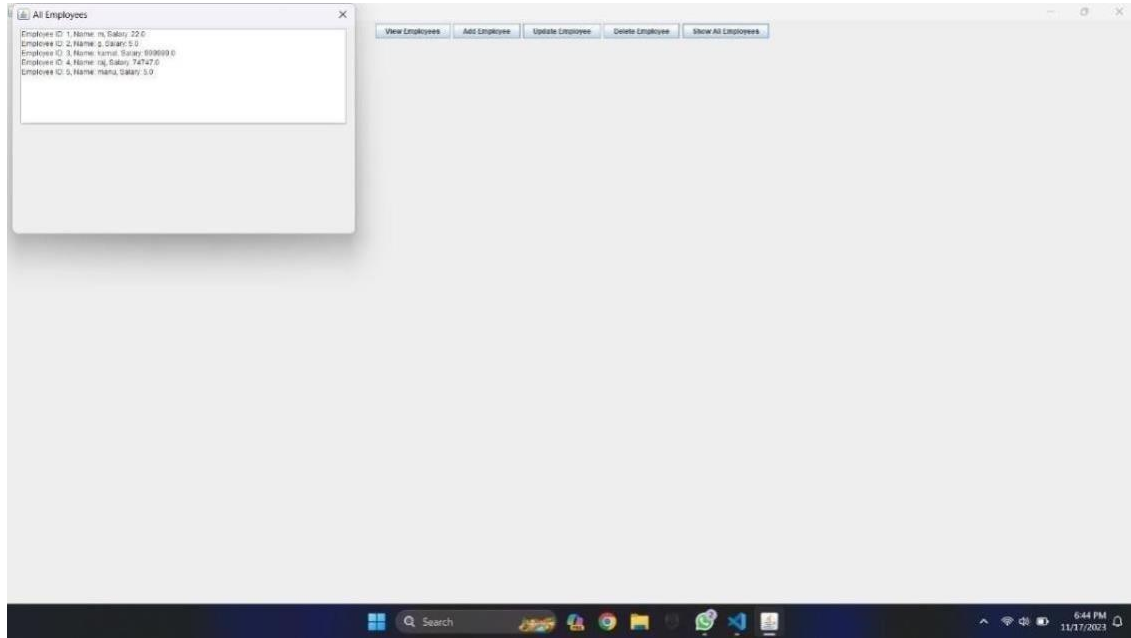
    e.printStackTrace();

    System.err.println("Error creating the table: " + e.getMessage()); }

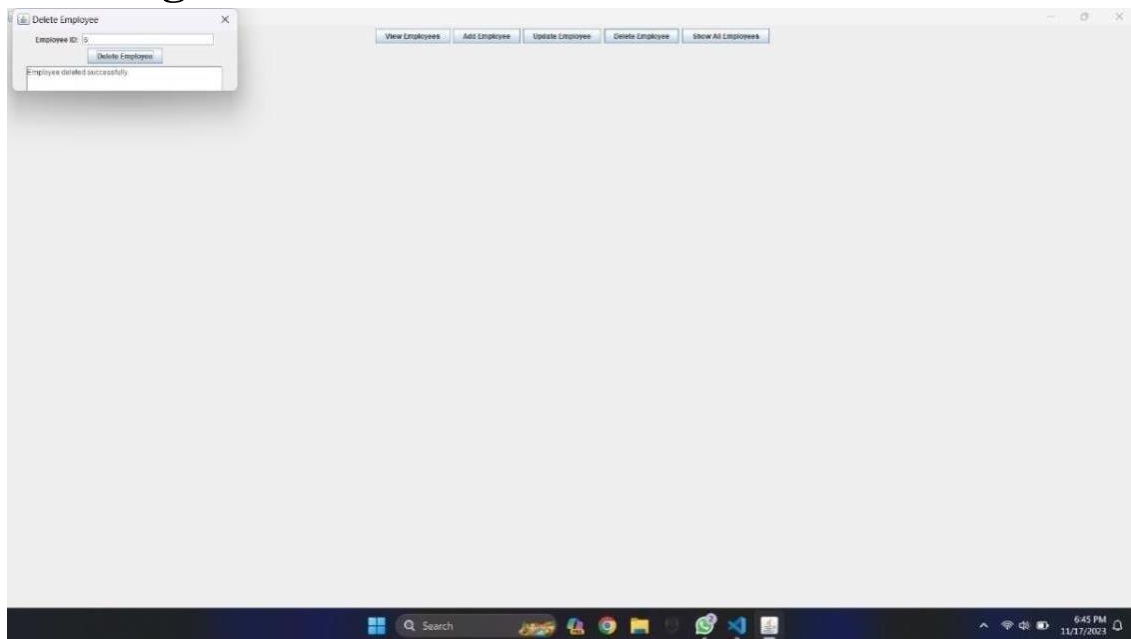
}
}

```

Viewing details



Deleting details



CONCLUSION:

The project titled as Employee Management System was deeply studied and analyzed to design the code and implement. It was done under the guidance of the experienced project guide. All the current requirements and possibilities have been taken care during the project time.

REFERENCE LINKS:

- [1] <https://www.javatpoint.com/java-awt>
 - [2] <https://www.javatpoint.com/java-swing>
-