

EMPLOYEE MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

RUPAVARTHINI K S 230701272

SHAKTHI G 230701303

In partial fulfillment for the award of the degree of

**BACHELOR OF
ENGINEERING IN
COMPUTER SCIENCE**



RAJALAKSHMI ENGINEERING COLLEGE

(AUTONOMOUS) THANDALAM

CHENNAI-602105

2024 - 2025

BONAFIDE CERTIFICATE

Certified that this project report “**EMPLOYEE MANAGEMENT SYSTEM**” is the bonafide work of “**RUPAVARTHINI K S (230701272), SHAKTHI G (230701303)**” who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE

**MRS V JANANEE
ASSISTANT PROFESSOR
Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai**

SIGNATURE

**MR SARAVANA GOKUL
ASSISTANT PROFESSOR
Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai**

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

The Employee Management System is a robust software solution designed to manage and streamline employee-related data, salary calculations, and payroll processing within an organization. This system provides an automated and efficient way to handle crucial HR tasks, reducing the time and errors associated with manual processes.

The system offers key functionalities such as employee management, where the admin can add, update, or remove employee details, ensuring that all records are up-to-date. It also facilitates attendance tracking, where the system records daily attendance, absences, and leave types, which directly influence salary calculations.

The salary calculation module allows the system to compute employees' salaries by considering factors like basic pay, allowances, and deductions, ensuring accurate and timely payments. Additionally, the system can generate payslips, which provide employees with a clear breakdown of their earnings and deductions.

An important feature of the system is the role-based access control, where different users (Admin, HR, Employees) have access to specific functionalities. This ensures that sensitive data is only accessible to authorized personnel. The system also maintains detailed records of taxes and deductions, which are vital for compliance with company policies and tax regulations.

Overall, this Employee Management System is designed to improve the efficiency and accuracy of payroll processing, providing organizations with a comprehensive solution for managing employee data and payroll operations.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1	INTRODUCTION	6
1.2	OBJECTIVES	7
1.3	MODULES	7

Chapter 2

SURVEY OF TECHNOLOGIES

1.4	SOFTWARE DESCRIPTION	9
1.5	LANGUAGES	9
1.5.1	PHP	9
1.5.2	SQL	9
1.5.3	HTML	10
1.5.4	CSS	10
1.5.5	JAVASCRIPT	10

Chapter 3

REQUIREMENTS AND ANALYSIS

1.6	REQUIREMENT SPECIFICATION	11
1.6.1	FUNCTIONAL REQUIREMENTS	11
1.6.2	NON FUNCTIONAL REQUIREMENTS	12
1.7	ER DIAGRAM	15
1.8	NORMALIZATION	16

Chapter 4

PROGRAM CODE

1.9	PROGRAM CODE	18
-----	--------------	----

Chapter 5

RESULTS AND DISCUSSION

1.10	RESULTS AND DISCUSSION	27
------	------------------------	----

Chapter 6

CONCLUSION

1.11 CONCLUSION	31
-----------------	----

Chapter 7

REFERENCES

1.12 REFERENCES	32
-----------------	----

Chapter 1

INTRODUCTION

The Employee Payroll Management System is a critical software solution designed to optimize and automate the management of employee data, attendance, salary calculations, and payroll processing. In any organization, maintaining accurate employee records and ensuring timely, error-free salary distribution is vital for smooth operations. This system aims to enhance the efficiency of HR departments by automating manual processes, minimizing errors, and ensuring transparency and consistency in payroll management.

The Employee Payroll Management System offers a comprehensive suite of features to meet the needs of both administrators and employees. The system allows for the easy management of employee records, including personal details, designation, salary information, and attendance history. Additionally, it facilitates the calculation of salaries based on attendance, allowances, deductions, and leave types, making the payroll process more accurate and efficient.

Developed using Java, MySQL, and Java Swing, this system leverages these technologies to provide a reliable, user-friendly platform. Java serves as the foundation for the system's frontend, providing a seamless interface for HR professionals and employees. MySQL is used for database management, ensuring smooth storage and retrieval of employee and payroll data. Java Swing is utilized for creating the graphical user interface (GUI), offering an intuitive and interactive user experience.

This report will outline the design, development process, and the technologies employed to create the Employee Payroll Management System, showcasing how the integration of these technologies leads to an efficient, reliable, and secure payroll management solution that ultimately enhances organizational operations and employee satisfaction.

1.1 OBJECTIVES

- To create a centralized database for managing employee records, attendance, and payroll data efficiently.
- To automate the calculation of salaries, deductions, allowances, and taxes for error-free payroll processing.
- To generate accurate payslips and comprehensive reports for both administrative and employee use.
- To provide a user-friendly interface for HR staff to manage employee details and payroll operations with ease.
- To ensure compliance with labor laws and organizational policies related to employee payments and deductions.
- To enhance coordination between HR, finance, and employees by offering real-time access to payroll and attendance data.

1.2 MODULES

1 Employee Management Module

The Employee Management Module is designed to handle all employee-related information efficiently. This module captures essential data, including employee names, contact information, job roles, departments, and hire dates. It provides a user-friendly interface for adding, updating, or removing employee details. By centralizing employee records, this module ensures quick access to information and accurate record-keeping, supporting smooth HR operations and compliance with organizational policies.

2 Attendance Tracking Module

The Attendance Tracking Module monitors and records employee attendance, absences, and leaves. It provides functionalities for marking daily attendance, specifying leave types, and tracking absenteeism. This module ensures real-time updates of attendance data, allowing administrators to generate accurate reports on employee punctuality and work hours. It also supports features for leave requests and approvals, streamlining attendance management.

3. Salary Calculation Module

The Salary Calculation Module automates the process of calculating salaries, deductions, and allowances for employees. It ensures accurate computation of net pay by considering basic salary, overtime, taxes, and other applicable deductions. This module simplifies payroll processing and reduces the chances of manual errors, enhancing efficiency and accuracy.

4. Payslip Generation Module

The Payslip Generation Module creates detailed payslips for employees, including breakdowns of earnings, deductions, and net salary. This module allows administrators to generate and distribute payslips automatically or on-demand. It supports exporting payslips in formats such as PDF, ensuring easy sharing and record-keeping for both employees and the organization.

5. Admin Dashboard Module

The Admin Dashboard Module serves as the central interface for managing the entire payroll system. It provides an overview of employee details, attendance records, salary disbursements, and tax deductions. The dashboard includes tools for generating reports, tracking payroll performance, and managing user roles. Its intuitive design allows administrators to perform tasks efficiently and access critical data with ease.

6. Tax and Deduction Module

The Tax and Deduction Module handles the computation and management of statutory deductions such as income tax, provident fund, and other organization-specific deductions. This module ensures compliance with tax regulations and provides administrators with detailed reports on employee contributions and deductions.

7. Database Module

The Database Module serves as the backbone of the Employee Payroll Management System. It securely stores and manages all data related to employees, attendance, salaries, deductions, and reports. Using a MongoDB database, this module supports efficient querying, data integrity, and scalability, enabling the system to handle large volumes of data seamlessly.

2.1 SOFTWARE DESCRIPTION

The Employee Payroll Management System employs a combination of technologies to ensure robust functionality and user-friendly operation. The backend is powered by a document-oriented database, while the frontend is designed for interactive and intuitive user experience. Middleware technologies provide seamless communication between the database and the user interface.

2.2 LANGUAGES

The system is developed using Java for the application logic, MongoDB for database management, and JavaFX or Servlets and JSP for the user interface. Additional libraries and tools enhance the application's efficiency and user experience.

2.2.1 Java

Role: Java is the primary programming language used for implementing the core logic and functionality of the Employee Payroll Management System.

Usage: Java handles backend processes, including data manipulation, salary computations, and business logic execution.

Advantages:

- Platform Independence: Java's "Write Once, Run Anywhere" capability allows the application to run on multiple platforms without modification.
- Robust Ecosystem: Java provides a vast library of APIs and frameworks, facilitating rapid development and scalability.

2.2.2 MySQL

Role: MySQL is the relational database management system used for storing and managing data in a structured and organized format.

Usage: Employee information, attendance records, salary details, and deductions are stored as tables in MySQL using a relational schema.

Advantages:

- Data Integrity: MySQL ensures data consistency and enforces relationships through constraints like primary keys and foreign keys.
- Efficiency: Supports optimized queries for efficient data retrieval and management, even with large datasets.
- Flexibility: Provides robust support for structured data and complex queries, adapting well to various requirements.

2.2.3 JavaFX/Servlets and JSP

Role: These technologies are used for building the graphical user interface (JavaFX) or dynamic web interfaces (Servlets and JSP).

Usage: JavaFX creates desktop applications with interactive UI components, while Servlets and JSP generate dynamic web pages for user interaction.

Advantages:

- **Rich UI Components:** JavaFX provides pre-built UI components for creating modern interfaces.
- **Dynamic Content:** Servlets and JSP enable real-time interaction with the server for data retrieval and updates.

2.2.4 HTML

Role: HTML (HyperText Markup Language) structures the content in the user interface, especially in the web-based version of the system.

Usage: HTML forms the backbone for designing forms, tables, and interactive content.

Advantages:

- **Universal Accessibility:** HTML is supported across all browsers, ensuring broad compatibility.
- **Integration:** It seamlessly integrates with CSS and JavaScript to enhance functionality and aesthetics.

2.2.5 CSS

Role: CSS (Cascading Style Sheets) styles the HTML elements for the web interface of the system.

Usage: CSS defines the visual layout, including colors, fonts, and spacing, for a consistent user experience.

Advantages:

- **Design Consistency:** Ensures uniform design across all pages and screens.
- **Responsive Design:** Adapts the interface for use on different devices and screen sizes.

2.2.6 JavaScript

Role: JavaScript (JS) adds interactivity and enhances user experience for web-based components.

Usage: Enables dynamic features such as validation of forms, dropdown interactions, and updating content without page reloads.

Advantages:

- **Interactivity:** Provides real-time feedback and dynamic user interactions.
- **Versatility:** Works across both client-side and server-side environments, ensuring comprehensive functionality.

3.1 REQUIREMENT SPECIFICATION

3.1.1 Functional Requirements

User Authentication and Authorization

- User Registration and Login: Allow users to register, create accounts, and log in securely.
- Role-Based Access Control: Assign specific permissions based on user roles (admin, HR, employee).

Employee Management

- Profile Creation and Update: Enable admins to add, update, and delete employee profiles.
- Employee Records: Maintain comprehensive records, including personal details and employment history.

Attendance Tracking

- Daily Attendance: Record attendance status for employees (present, absent, leave).
- Leave Management: Track leave types (casual, sick, etc.) and balances for employees.

Salary Processing

- Payroll Calculation: Compute monthly salaries based on attendance, deductions, and allowances.
- Payslip Generation: Generate detailed payslips for

employees. Admin Dashboard Module

- Comprehensive Overview: Provide an overview of employee records, attendance summaries, and payroll status.
- Management Tools: Include reporting tools for salary trends, tax deductions, and employee statistics.

Reporting and Analytics

- Custom Reports: Generate reports based on employee data, attendance, or payroll.
- Data Export: Allow data export in formats like PDF or Excel.

3.1.2 Non-Functional Requirements

Security

- **Data Encryption:** Encrypt sensitive data both in transit and at rest.
- **Role-Based Access:** Ensure secure access based on user

Performance

- **Scalability:** Design the system to handle a growing number of employees and data efficiently.
- **Response Time:** Ensure prompt responses to user actions, such as payroll calculations or report generation.

Reliability

- **Availability:** Guarantee high system availability with minimal downtime.
- **Data Backup:** Implement regular backups to prevent data loss and support recovery.

Usability

- **User-Friendly Interface:** Provide an intuitive and easy-to-navigate interface for all users.
- **Accessibility:** Ensure accessibility for users with disabilities, adhering to relevant standards.

Maintainability

- **Modular Design:** Employ a modular architecture to simplify maintenance and upgrades.
- **Documentation:** Include comprehensive documentation for developers and end-users.

Interoperability

- **System Integration:** Support integration with other systems like tax filing, financial software, or HR systems.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

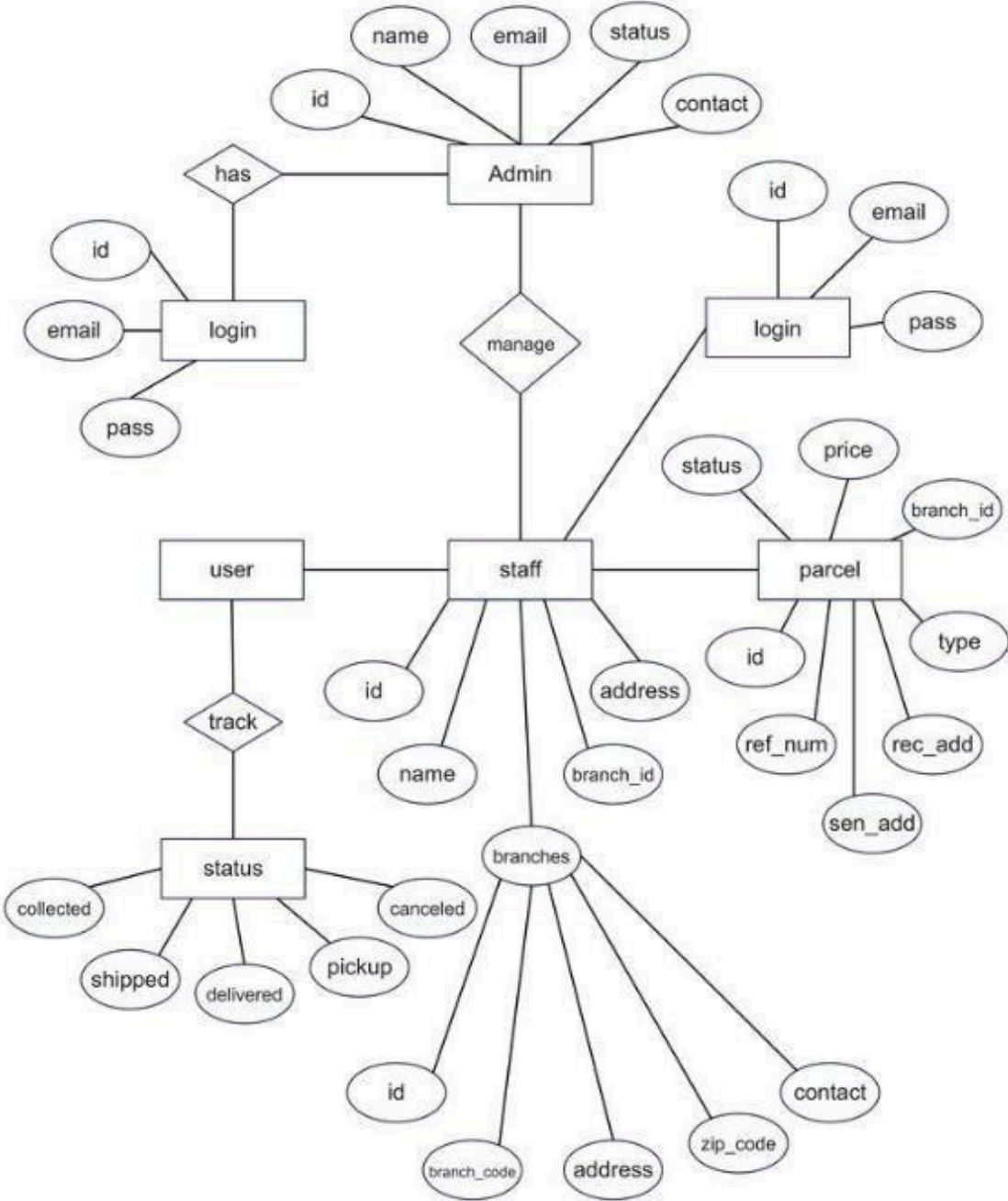
Hardware Requirements

- Desktop PC or Laptop: A reliable machine to host the Employee Payroll Management System.
- Processor: Intel® Core TM i3-6006U CPU @ 2.00GHz or equivalent for smooth processing.
- RAM: 4.00 GB RAM to manage concurrent users and database operations efficiently.
- System Architecture: 64-bit operating system, x64-based processor for better performance.
- Monitor Resolution: 1024 x 768 or higher resolution for a clear display of the user interface.
- Input Devices: Keyboard and mouse for user interaction.
- Server: High processing power and ample storage capacity.
- Network: Reliable network infrastructure for seamless

operation. Software Requirements

- Operating System: Windows 10 or higher.
- Code Editor: IntelliJ IDEA, Eclipse, or NetBeans.
- Front End: JavaSwing.
- Back End: MySQL.
- Database: MySQL
- Reporting Tools: JasperReports or iText.
- Version Control: Git.

ER Diagram :



3.1 NORMALISATION

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing a database into two or more tables and defining relationships between the tables. The steps to normalize a database table for Employee Management System are as follows.

Raw Database

Attribute	Datatype	Example Value
name	VARCHAR(50)	HARITHA
fname	VARCHAR(50)	MALLAIAN
address	VARCHAR(50)	CHROMEPET
Phone_number	NUMBER(10,2)	9785044002
Emp_ID	VARCHAR(50)	HR234
DOB	Date	10/11/2002
Designation	VARCHAR(50)	HR
Aadhar	VARCHAR(50)	9837 6788 5677
Education	VARCHAR(50)	B.Tech

1. LOGIN PAGE

```
package employee.payroll.management;

import javax.swing.*;

import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class Login extends JFrame implements ActionListener {

    JTextField tfusername, tfpassword;

    Login() {
        getContentPane().setBackground(Color.WHITE);
        setLayout(null);

        JLabel lblusername = new JLabel("Username");
        lblusername.setBounds(40, 35, 100, 30);
        add(lblusername);

        tfusername = new JTextField();
        tfusername.setBounds(160, 35, 150, 30);
        add(tfusername);

        JLabel lblpassword = new JLabel("Password");
        lblpassword.setBounds(40, 90, 100, 30);
        add(lblpassword);

        tfpassword = new JTextField();
        tfpassword.setBounds(160, 90, 150, 30);
        add(tfpassword);

        JButton clickhere = new JButton("LOGIN");
        clickhere.setBounds(160, 145, 150, 35);
        clickhere.setBackground(Color.BLACK);
        clickhere.setForeground(Color.WHITE);
        clickhere.addActionListener(this); // Add action listener for the
        button add(clickhere);
```



```

// Load image as a resource
ImageIcon i1 = null;
try {
    i1 = new ImageIcon(getClass().getResource("/user.jpg"));
} catch (Exception e) {
    System.out.println("Image not found: " + e.getMessage());
}

if (i1 != null) {
    Image i2 = i1.getImage().getScaledInstance(150, 100, Image.SCALE_DEFAULT);
    ImageIcon i3 = new ImageIcon(i2);
    JLabel image = new JLabel(i3);
    image.setBounds(370, 50, 160, 120); // Adjusted bounds to fit within frame
    add(image);
} else {
    System.out.println("Error: Image not loaded from classpath.");
}

setSize(600, 300);
setLocation(450, 200);
setVisible(true);
}

@Override
public void actionPerformed(ActionEvent ae) {
    try {
        String username = tfusername.getText();
        String password = tfpassword.getText();

        Conn c = new Conn();

        String query = "select * from login where username ='" + username + "' and password = '" +
password + "'";
        ResultSet rs = c.s.executeQuery(query);
        if (rs.next()) {
            setVisible(false);
            new Home();
        } else {
            JOptionPane.showMessageDialog(null, "Invalid Username or password");
            setVisible(false);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```
}

// Handle button click here
System.out.println("Button clicked!");
}

public static void main(String[] args) {
    new Login();
}
```

2.HOME PAGE

```
package
employee.payroll.management; import
java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Home extends JFrame implements ActionListener {
    JComboBox<String> actionComboBox;
    JButton goButton;

    Home() {
        setLayout(null)
        ;

        // Load image from an absolute path
        ImageIcon i1 = new ImageIcon("C:\\Users\\Shanmuga
Priya\\Documents\\NetBeansProjects\\Employee Payroll Management\\src\\image.jpg");

        if (i1.getImageLoadStatus() == MediaTracker.COMPLETE) {
            Image i2 = i1.getImage().getScaledInstance(1100, 700, Image.SCALE_DEFAULT);
            ImageIcon i3 = new ImageIcon(i2);
            JLabel image = new JLabel(i3);
            image.setBounds(0, 0, 1120, 630);
            add(image);

            JLabel heading = new JLabel("Employee Payroll Management");
            heading.setBounds(310, 20, 500, 40);
            heading.setFont(new Font("Tahoma", Font.BOLD, 25));
            image.add(heading);

            // Create ComboBox with the actions
            String[] actions = { "Select Action", "Add Employee", "View Employee", "Update
Employee", "Remove Employee", "Payroll", "Attendance" };
            actionComboBox = new
            JComboBox<>(actions);
            actionComboBox.setBounds(350, 100, 200, 40);
            image.add(actionComboBox);

            // Create 'Go' button to perform the selected
            action goButton = new JButton("Go");
            goButton.setBounds(550, 100, 100, 40);
            goButton.addActionListener(this);
            image.add(goButton);

        } else {
            System.out.println("Error: Image not loaded from specified path.");
        }

        setSize(1120, 620);
```

```

        setLocation(250,
        100); setVisible(true);
    }

    public void actionPerformed(ActionEvent ae) {
        if (ae.getSource() == goButton) {
            String selectedAction = (String) actionComboBox.getSelectedItem();

            if (selectedAction == null || selectedAction.equals("Select Action")) {
                JOptionPane.showMessageDialog(this, "Please select a valid
                action."); return;
            }

            // Open the respective page based on the
            selection switch (selectedAction) {
                case "Add Employee":
                    new AddEmployee(); // Open the Add Employee page
                    break;
                case "View Employee":
                    new ViewEmployee(); // Open the View Employee page
                    break;
                case "Update Employee":
                    new UpdateEmployee(); // Open the Update Employee page
                    break;
                case "Remove Employee":
                    this.setVisible(false); // Hide the Home page
                    new RemoveEmployee(); // Open the Remove Employee page
                    break;
                case "Payroll":
                    new Payroll(); // Open the Payroll page
                    break;
                case "Attendance":
                    new Attendance(); // Open the Attendance page (Make sure this class exists)
                    break;
                default:
                    JOptionPane.showMessageDialog(this, "Invalid
                    selection!"); break;
            }
        }
    }

    public static void main(String[] args) {
        new Home(); // Launch the Home page
    }

    private static class Attendance {

        public Attendance() {
        }
    }
}

```

3.ADD EMPLOYEE PAGE

```
package
employee.payroll.management; import
javax.swing.*;
import javax.swing.border.Border;
import javax.swing.border.LineBorder;
import javax.swing.event.DocumentEvent;
import
javax.swing.event.DocumentListener;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import com.toedter.calendar.JDateChooser;

public class AddEmployee extends JFrame implements ActionListener {
    // Declare text fields and components
    JTextField tfname, tffname, tfaddress, tfphone, tfaadhar, tfemail, tfSalary, tfDesignation,
    tfEmpId;
    JComboBox<String> educationComboBox;
    JButton submitButton, cancelButton;
    JDateChooser dcdob;

    // Borders for validation
    Border greenBorder = new LineBorder(Color.GREEN, 2);
    Border redBorder = new LineBorder(Color.RED, 2);
    Border defaultBorder = new JTextField().getBorder();

    AddEmployee() {
        getContentPane().setBackground(Color.WHITE);
        setLayout(null);

        // Heading
        JLabel heading = new JLabel("Add Employee Detail");
        heading.setBounds(320, 30, 500, 50);
        heading.setFont(new Font("SAN_SERIF", Font.BOLD,
        25)); add(heading);

        // Name
        JLabel labelname = new JLabel("Name");
        labelname.setBounds(50, 150, 150, 30);
        labelname.setFont(new Font("serif", Font.PLAIN, 20));
        add(labelname);

        tfname = new JTextField();
        tfname.setBounds(200, 150, 150, 30);
        add(tfname);

        // Father's Name
        JLabel labelfname = new JLabel("Father's Name");
        labelfname.setBounds(400, 150, 150, 30);
```

```

labelfname.setFont(new Font("serif", Font.PLAIN,
20)); add(labelfname);

tffname = new JTextField();
tffname.setBounds(600, 150, 150, 30);
add(tffname);

// Date of Birth
JLabel labeldob = new JLabel("Date of Birth");
labeldob.setBounds(50, 200, 150, 30);
labeldob.setFont(new Font("serif", Font.PLAIN,
20)); add(labeldob);

dcdob = new JDateChooser();
dcdob.setBounds(200, 200, 150,
30); add(dcdob);

// Salary
JLabel labelSalary = new JLabel("Salary");
labelSalary.setBounds(400, 200, 150, 30);
labelSalary.setFont(new Font("serif", Font.PLAIN, 20));
add(labelSalary);

tfSalary = new JTextField();
tfSalary.setBounds(600, 200, 150, 30);
add(tfSalary);

// Address
JLabel labeladdress = new JLabel("Address");
labeladdress.setBounds(50, 250, 150, 30);
labeladdress.setFont(new Font("serif", Font.PLAIN,
20)); add(labeladdress);

tfaddress = new JTextField();
tfaddress.setBounds(200, 250, 550, 30);
add(tfaddress);

// Phone Number
JLabel labelPhone = new JLabel("Phone");
labelPhone.setBounds(50, 300, 150, 30);
labelPhone.setFont(new Font("serif", Font.PLAIN,
20)); add(labelPhone);

tfphone = new JTextField();
tfphone.setBounds(200, 300, 150, 30);
tfphone.getDocument().addDocumentListener(new InputValidationListener(tfphone, 10,
true)); // Add dynamic validation
add(tfphone);

// Email
JLabel labelEmail = new JLabel("Email");
labelEmail.setBounds(400, 300, 150, 30);

```

```

labelEmail.setFont(new Font("serif", Font.PLAIN,
20)); add(labelEmail);

tfemail = new JTextField();
tfemail.setBounds(600, 300, 150,
30); add(tfemail);

// Education
JLabel labelEducation = new JLabel("Education");
labelEducation.setBounds(50, 350, 150, 30);
labelEducation.setFont(new Font("serif", Font.PLAIN,
20)); add(labelEducation);

String[] educationOptions = {"BBA", "BCA", "BA", "B.COM", "BTECH", "MBA", "MCA",
"BE", "ME"};
educationComboBox = new JComboBox<>(educationOptions);
educationComboBox.setBounds(200, 350, 150, 30);
add(educationComboBox);

// Designation
JLabel labelDesignation = new JLabel("Designation");
labelDesignation.setBounds(400, 350, 150, 30);
labelDesignation.setFont(new Font("serif", Font.PLAIN, 20));
add(labelDesignation);

tfDesignation = new JTextField();
tfDesignation.setBounds(600, 350, 150,
30); add(tfDesignation);

// Aadhar Number
JLabel labelAadhar = new JLabel("Aadhar Number");
labelAadhar.setBounds(50, 400, 150, 30);
labelAadhar.setFont(new Font("serif", Font.PLAIN, 20));
add(labelAadhar);

tfaadhar = new JTextField();
tfaadhar.setBounds(200, 400, 150, 30);
tfaadhar.getDocument().addDocumentListener(new InputValidationListener(tfaadhar, 12,
true)); // Add dynamic validation
add(tfaadhar);

// Employee ID
JLabel labelEmpId = new JLabel("Employee ID");
labelEmpId.setBounds(400, 400, 150, 30);
labelEmpId.setFont(new Font("serif", Font.PLAIN, 20));
add(labelEmpId);

tfEmpId = new JTextField();
tfEmpId.setBounds(600, 400, 150,
30); add(tfEmpId);

// Submit Button

```

```

submitButton = new JButton("Submit");
submitButton.setBounds(250, 500, 150, 40);
submitButton.setBackground(Color.BLACK);
submitButton.setForeground(Color.WHITE);
submitButton.addActionListener(this);
add(submitButton);

// Cancel Button
cancelButton = new JButton("Cancel");
cancelButton.setBounds(450, 500, 150, 40);
cancelButton.setBackground(Color.BLACK);
cancelButton.setForeground(Color.WHITE);
cancelButton.addActionListener(this);
add(cancelButton);

setSize(900, 700);
setLocation(300,
50); setVisible(true);
}

// Input validation listener for dynamic feedback
class InputValidationListener implements DocumentListener {
    private final JTextField textField;
    private final int requiredLength;
    private final boolean onlyDigits;

    public InputValidationListener(JTextField textField, int requiredLength, boolean onlyDigits)
    {
        this.textField = textField;
        this.requiredLength =
        requiredLength; this.onlyDigits =
        onlyDigits;
    }

    @Override
    public void insertUpdate(DocumentEvent e) {
        validateInput();
    }

    @Override
    public void removeUpdate(DocumentEvent e)
    { validateInput();
    }

    @Override
    public void changedUpdate(DocumentEvent e) {
        validateInput();
    }

    private void validateInput() {
        String text = textField.getText();
        boolean isValid = text.length() == requiredLength && (!onlyDigits || text.matches("\\d+"));
    }
}

```



```

        if (isValid) {
            textField.setBorder(greenBorder);
        } else if (text.isEmpty()) {
            textField.setBorder(defaultBorder);
        };
    } else {
        textField.setBorder(redBorder);
    }
}
}
}

```

@Override

```

public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == submitButton) {
        // Get values from form fields
        String name = tfname.getText();
        String fname = tffname.getText();
        String dob = ((JTextField) dcdob.getDateEditor().getUiComponent()).getText();
        String address = tfaddress.getText();
        String phone = tfphone.getText();
        String email = tfemail.getText();
        String education = (String) educationComboBox.getSelectedItem();
        String designation = tfDesignation.getText();
        String aadhar = tfaadhar.getText();
        String empId = tfEmpId.getText();

        // Validate phone and aadhar before submitting
        if (phone.length() != 10 || aadhar.length() != 12 || !phone.matches("\\d+") ||
!aadhar.matches("\\d+")) {
            JOptionPane.showMessageDialog(this, "Please enter valid phone (10 digits) and Aadhar
(12 digits) numbers.");
            return;
        }

        // Proceed with database insertion
        logic try {
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/employeemanagementsystem", "root",
"Pradanya#2011");

            String query = "INSERT INTO employee (name, fname, dob, address, phone, email,
education, designation, aadhar, empID) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
            PreparedStatement stmt =
conn.prepareStatement(query); stmt.setString(1, name);
            stmt.setString(2, fname);
            stmt.setString(3, dob);
            stmt.setString(4, address);
            stmt.setString(5, phone);
            stmt.setString(6, email);
            stmt.setString(7, education);
            stmt.setString(8, designation);
            stmt.setString(9, aadhar);
            stmt.setString(10, empId);

```

```

        int rowsInserted = stmt.executeUpdate();
        if (rowsInserted > 0) {
            JOptionPane.showMessageDialog(this, "Employee added successfully.");
            setVisible(false);
            new Home(); // Assuming Home is another JFrame class for the home screen
        }
        conn.close();
    } catch (Exception e)
    {
        e.printStackTrace();
    }
    } else if (ae.getSource() == cancelButton)
    { setVisible(false);
    }
}

public static void main(String[] args) {
    new AddEmployee();
}
}

```

4.UPDATE EMPLOYEE PAGE

```
package
employee.payroll.management; import
javax.swing.*;
import javax.swing.border.Border;
import javax.swing.border.LineBorder;
import com.toedter.calendar.JDateChooser;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class UpdateEmployee extends JFrame implements ActionListener {
    // Declare text fields and components
    JTextField tfname, tffname, tfaddress, tfphone, tfaadhar, tfemail, tfSalary, tfDesignation,
    tfEmpId;
    JComboBox<String> educationComboBox;
    JButton updateButton, searchButton, backButton;
    JDateChooser dcdob;

    // Borders for validation
    Border greenBorder = new LineBorder(Color.GREEN, 2);
    Border redBorder = new LineBorder(Color.RED, 2);
    Border defaultBorder = new JTextField().getBorder();

    // Constructor
    public UpdateEmployee() {
        getContentPane().setBackground(Color.WHITE);
        setLayout(null);

        // Heading
        JLabel heading = new JLabel("Update Employee
        Detail"); heading.setBounds(320, 30, 500, 50);
        heading.setFont(new Font("SAN_SERIF", Font.BOLD,
        25)); add(heading);

        // Employee ID Label and TextField
        JLabel labelEmpId = new JLabel("Employee ID");
        labelEmpId.setBounds(50, 100, 150, 30);
        labelEmpId.setFont(new Font("serif", Font.PLAIN, 20));
        add(labelEmpId);

        tfEmpId = new JTextField();
        tfEmpId.setBounds(200, 100, 150,
        30); add(tfEmpId);

        // Search Button
        searchButton = new JButton("Search");
        searchButton.setBounds(370, 100, 100, 30);
        searchButton.addActionListener(this);
```

```
add(searchButton);

// Name Label and TextField
JLabel labelName = new JLabel("Name");
labelName.setBounds(50, 150, 150, 30);
labelName.setFont(new Font("serif", Font.PLAIN,
20)); add(labelName);

tfname = new JTextField();
tfname.setBounds(200, 150, 150, 30);
tfname.setEditable(false);
add(tfname);

// Father's Name Label and TextField
JLabel labelFname = new JLabel("Father's Name");
labelFname.setBounds(400, 150, 150, 30);
labelFname.setFont(new Font("serif", Font.PLAIN, 20));
add(labelFname);

tffname = new JTextField();
tffname.setBounds(600, 150, 150, 30);
tffname.setEditable(false);
add(tffname);

// Date of Birth Label and JDateChooser
JLabel labeldob = new JLabel("Date of
Birth"); labeldob.setBounds(50, 200, 150, 30);
labeldob.setFont(new Font("serif", Font.PLAIN,
20)); add(labeldob);

dcdob = new JDateChooser();
dcdob.setBounds(200, 200, 150,
30); dcdob.setEnabled(false);
add(dcdob);

// Address Label and TextField
JLabel labeladdress = new JLabel("Address");
labeladdress.setBounds(400, 200, 150, 30);
labeladdress.setFont(new Font("serif", Font.PLAIN,
20)); add(labeladdress);

tfaddress = new JTextField();
tfaddress.setBounds(600, 200, 150, 30);
tfaddress.setEditable(false);
add(tfaddress);

// Phone Label and TextField
JLabel labelPhone = new JLabel("Phone");
labelPhone.setBounds(50, 250, 150, 30);
labelPhone.setFont(new Font("serif", Font.PLAIN,
20)); add(labelPhone);
```

```

tfphone = new JTextField();
tfphone.setBounds(200, 250, 150, 30);
tfphone.setEditable(false);
add(tfphone);

// Email Label and TextField
JLabel labelEmail = new JLabel("Email");
labelEmail.setBounds(400, 250, 150, 30);
labelEmail.setFont(new Font("serif", Font.PLAIN,
20)); add(labelEmail);

tfemail = new JTextField();
tfemail.setBounds(600, 250, 150,
30); tfemail.setEditable(false);
add(tfemail);

// Salary Label and TextField
JLabel labelSalary = new JLabel("Salary");
labelSalary.setBounds(50, 300, 150, 30);
labelSalary.setFont(new Font("serif", Font.PLAIN, 20));
add(labelSalary);

tfSalary = new JTextField();
tfSalary.setBounds(200, 300, 150, 30);
tfSalary.setEditable(false);
add(tfSalary);

// Education Label and ComboBox
JLabel labelEducation = new JLabel("Education");
labelEducation.setBounds(400, 300, 150, 30);
labelEducation.setFont(new Font("serif", Font.PLAIN,
20)); add(labelEducation);

String[] educationOptions = {"BBA", "BCA", "BA", "B.COM", "BTECH", "MBA", "MCA",
"BE", "ME"};
educationComboBox = new JComboBox<>(educationOptions);
educationComboBox.setBounds(600, 300, 150, 30);
educationComboBox.setEnabled(false);
add(educationComboBox);

// Designation Label and TextField
JLabel labelDesignation = new JLabel("Designation");
labelDesignation.setBounds(50, 350, 150, 30);
labelDesignation.setFont(new Font("serif", Font.PLAIN, 20));
add(labelDesignation);

tfDesignation = new JTextField();
tfDesignation.setBounds(200, 350, 150,
30); tfDesignation.setEditable(false);
add(tfDesignation);

// Aadhar Number Label and TextField

```

```
JLabel labelAadhar = new JLabel("Aadhar Number");
labelAadhar.setBounds(400, 350, 150, 30);
labelAadhar.setFont(new Font("serif", Font.PLAIN, 20));
add(labelAadhar);
```

```
tfaadhar = new JTextField();
tfaadhar.setBounds(600, 350, 150, 30);
tfaadhar.setEditable(false);
add(tfaadhar);
```

```
// Back Button
backButton = new JButton("Back");
backButton.setBounds(450, 500, 150, 40);
backButton.setBackground(Color.BLACK);
backButton.setForeground(Color.WHITE);
backButton.addActionListener(this);
add(backButton);
```

```
setSize(900, 600);
setLocation(300,
100); setVisible(true);
}
```

```
UpdateEmployee(String selectedEmpId) {
    throw new UnsupportedOperationException("Not supported yet."); // Generated
from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
}
```

```
// Search Employee Details
```

```
private void searchEmployeeById(String empId)
{ try {
    Conn conn = new Conn();
    String query = "SELECT * FROM employee WHERE empID =
?"; PreparedStatement pst = conn.c.prepareStatement(query);
    pst.setString(1, empId);
    ResultSet rs = pst.executeQuery();

    if (rs.next()) {
        tfname.setText(rs.getString("name"));
        tffname.setText(rs.getString("fname"));
        dcdob.setDate(rs.getDate("dob"));
        tfaddress.setText(rs.getString("address"));
        tfphone.setText(rs.getString("phone"));
        tfemail.setText(rs.getString("email"));
        tfSalary.setText(rs.getString("salary"));
        educationComboBox.setSelectedItem(rs.getString("education"));
        tfDesignation.setText(rs.getString("designation"));
        tfaadhar.setText(rs.getString("aadhar"));
        JOptionPane.showMessageDialog(null, "Employee details fetched successfully!");
    } else {
        JOptionPane.showMessageDialog(null, "No record found.");
    }
}
```

```

        pst.close();
        conn.c.close();
    } catch (Exception e)
    {
        e.printStackTrace();
        JOptionPane.showMessageDialog(null, "Database error: " + e.getMessage());
    }
}

@Override
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == searchButton) {
        String empId = tfEmpId.getText().trim();
        if (empId.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter Employee ID."); return;
        }
        searchEmployeeById(empId);
    } else if (ae.getSource() == backButton) {
        this.setVisible(false);
        new Home();
    }
}

public static void main(String[] args) {
    new UpdateEmployee();
}
}

```

5. IEW EMPLOYEE

```
package employee.payroll.management;
```

```
import javax.swing.*;
```

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
import java.sql.*;
```

```
import net.proteanit.sql.DbUtils;
```

```
public class ViewEmployee extends JFrame implements ActionListener {
```

```
    JTable table;
```

```
    JComboBox<String> employeeId; // Changed to JComboBox for better  
    functionality
```

```
    JButton searchButton, resetButton, backButton;
```

```
    ViewEmployee() {
```

```
        // Frame settings
```

```
        setTitle("View Employee Details");
```

```
        setSize(900, 700);
```

```
        setLocation(300, 100);
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        getContentPane().setBackground(Color.WHITE);
```

```
        setLayout(null);
```

```
        // Label for search by Employee ID
```

```
        JLabel searchLabel = new JLabel("Search by Employee ID:");
```

```
        searchLabel.setBounds(20, 20, 150, 20);
```

```
        add(searchLabel);
```

```
        // ComboBox for employee IDs
```

```
        employeeId = new JComboBox<>();
```

```
        employeeId.setBounds(180, 20, 150, 20);
```

```
        add(employeeId);
```

```
        // Populate the employee ID combo
```

```
        box try {
```

```
            Conn c = new Conn();
```

```
            ResultSet rs = c.s.executeQuery("SELECT empID FROM  
            employee"); while (rs.next()) {
```

```
                employeeId.addItem(rs.getString("empID"));
```

```
            }
```



```

    } catch (Exception e)
    {
        e.printStackTrace();
    }

    // Table to display employee data
    table = new JTable();
    loadTableData(); // Load initial data for all employees

    // Scroll pane for table
    JScrollPane jsp = new
    JScrollPane(table); jsp.setBounds(0, 100,
    900, 600); add(jsp);

    // Search button
    searchButton = new JButton("Search");
    searchButton.setBounds(350, 20, 100, 20);
    searchButton.addActionListener(this);
    add(searchButton);

    // Reset button to load all employee data
    resetButton = new JButton("Reset");
    resetButton.setBounds(470, 20, 100, 20);
    resetButton.addActionListener(this);
    add(resetButton);

    // Back button to return to the home page
    backButton = new JButton("Back");
    backButton.setBounds(590, 20, 100, 20);
    backButton.addActionListener(this);
    add(backButton);

    setVisible(true);
}

// Load all employee data into the table
private void loadTableData() {
    try {
        Conn c = new Conn();
        ResultSet rs = c.s.executeQuery("SELECT * FROM employee");
        table.setModel(DbUtils.resultSetToTableModel(rs));
    } catch (Exception e)
    {
        e.printStackTrace();
    }
}

```

```

    }

    // Search for a specific employee by empID
    private void searchEmployeeById(String empID) {
        try {
            Conn c = new Conn();
            String query = "SELECT * FROM employee WHERE empID =
?"; PreparedStatement pst = c.c.prepareStatement(query);
            pst.setString(1, empID);
            ResultSet rs = pst.executeQuery();
            if (rs.next()) {
                // Set the model of the table with the result
                table.setModel(DbUtils.resultSetToTableModel(rs));
            } else {
                JOptionPane.showMessageDialog(null, "No employee found with ID: "
+ empID);
            }
        } catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    // Handle button actions
    public void actionPerformed(ActionEvent ae) {
        if (ae.getSource() == searchButton) {
            String selectedEmpId = (String) cemployeeId.getSelectedItemAt(0); // Get the
selected employee ID
            searchEmployeeById(selectedEmpId);
        } else if (ae.getSource() == resetButton) {
            loadTableData(); // Reload all employee data
        } else if (ae.getSource() == backButton) {
            setVisible(false); // Close the current frame new
            HomePage(); // Open the home page frame
        }
    }

    public static void main(String[] args) {
        new ViewEmployee();
    }

    private static class HomePage {
        public HomePage() {
            JOptionPane.showMessageDialog(null, "Returning to Home Page");
        }
    }
}

```

6. REMOVE EMPLOYEE PAGE

```
package employee.payroll.management;

import javax.swing.*.*;
import java.awt.*.*;
import
java.awt.event.*;
import java.sql.*;

public class RemoveEmployee extends JFrame implements
    ActionListener { JComboBox<String> cEmpId; // JComboBox to
    display employee IDs JButton deleteButton, cancelButton;

    RemoveEmployee() {
        getContentPane().setBackground(Color.WHIT
        E); setLayout(null);

        // Heading
        JLabel heading = new JLabel("Remove
        Employee"); heading.setBounds(320, 30, 500,
        50);
        heading.setFont(new Font("SAN_SERIF", Font.BOLD, 25));
        add(heading);

        // Employee ID label
        JLabel labelEmpId = new JLabel("Employee ID");
        labelEmpId.setBounds(50, 150, 150, 30);
        labelEmpId.setFont(new Font("serif", Font.PLAIN, 20));
        add(labelEmpId);

        // JComboBox to select Employee
        ID cEmpId = new JComboBox<>();
        cEmpId.setBounds(200, 150, 150,
        30); add(cEmpId);

        // Populate JComboBox with employee IDs from the
        database try {
            Conn c = new Conn();
            ResultSet rs = c.s.executeQuery("SELECT empID FROM
            employee"); while (rs.next()) {
                cEmpId.addItem(rs.getString("empID"));
            }
        } catch (SQLException e)
        { e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Database error: " + e.getMessage());
```

```

    }

    // Delete Button
    deleteButton = new JButton("Delete");
    deleteButton.setBounds(250, 250, 150,
    40);
    deleteButton.setBackground(Color.BLAC
    K);
    deleteButton.setForeground(Color.WHITE
    ); deleteButton.addActionListener(this);
    add(deleteButton);

    // Cancel Button
    cancelButton = new JButton("Cancel");
    cancelButton.setBounds(450, 250, 150,
    40);
    cancelButton.setBackground(Color.BLAC
    K);
    cancelButton.setForeground(Color.WHITE
    ); cancelButton.addActionListener(this);
    add(cancelButton);

    setSize(900, 700);
    setLocation(300, 50);
    setVisible(true);
}

@Override
public void actionPerformed(ActionEvent
    ae) { if (ae.getSource() == deleteButton) {
        String empId = (String) cEmpId.getSelectedItem();

        if (empId == null) {
            JOptionPane.showMessageDialog(this, "Please select an Employee ID.");
            return;
        }

        // Ask for confirmation before deletion
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete
employee ID: " + empId + "?", "Confirm Deletion", JOptionPane.YES_NO_OPTION);
        if (confirm == JOptionPane.YES_OPTION) {
            // Database connection and deletion
            logic try {
                // Connect to the
                database Connection
                conn =
                DriverManager.getConnection("jdbc:mysql://localhost:3306/employeemanagementsystem", "root",

```

"Pradanya#2011"

```

// Create SQL delete query
String query = "DELETE FROM employee WHERE empID = ?";

// Prepare the statement
PreparedStatement pst = conn.prepareStatement(query);
pst.setString(1, empId);

// Execute the delete
int rowsAffected = pst.executeUpdate();

if (rowsAffected > 0)
{
    JOptionPane.showMessageDialog(this, "Employee removed
                                   successfully!");
} else
{
    JOptionPane.showMessageDialog(this, "No employee found with the given ID.");
}

// Close the connection
pst.close();
conn.close();

} catch (SQLException e)
{
    e.printStackTrace();
    JOptionPane.showMessageDialog(this, "Database error: " + e.getMessage());
}

} else if (ae.getSource() == cancelButton) {
    setVisible(false);
    new Home(); // Redirect to Home page or frame
}

}

public static void main(String[]
    args) { new RemoveEmployee();
}

}

```

7. PAYROLL PAGE

```
package employee.payroll.management;

import javax.swing.*.*;
import java.awt.*.*;
import
java.awt.event.*;
import java.sql.*.*;

public class Payroll extends JFrame implements ActionListener

    { Choice cemployeeId, cType;

    JTextField tfPercentage;
    JButton calculate, apply,
    back; JLabel resultLabel;

    Payroll() {
        // Frame settings
        setTitle("Calculate Bonus or Increment");
        setSize(500, 400);
        setLocation(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE
        ); setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        // Employee ID label and dropdown
        JLabel empIdLabel = new JLabel("Select Employee
        ID:"); empIdLabel.setBounds(50, 50, 150, 25);
        add(empIdLabel);

        cemployeeId = new Choice();
        cemployeeId.setBounds(220, 50, 150, 25);
        add(cemployeeId);

        // Populate employee
        IDs try {
            Conn c = new Conn();
            ResultSet rs = c.s.executeQuery("SELECT empID FROM
            employee"); while (rs.next()) {
                cemployeeId.add(rs.getString("empID"));
            }
        } catch (Exception e)
        {
            e.printStackTrace();
        }

        // Type (Bonus or Increment) label and dropdown
        JLabel typeLabel = new JLabel("Select Type:");
        typeLabel.setBounds(50, 100, 150, 25);
```

```

add(typeLabel);

cType = new Choice();
cType.add("Bonus");
cType.add("Increment");
cType.setBounds(220, 100, 150, 25);
add(cType);

// Percentage input
JLabel percentageLabel = new JLabel("Enter
Percentage:"); percentageLabel.setBounds(50, 150, 150,
25); add(percentagelabel);

tfPercentage = new JTextField();
tfPercentage.setBounds(220, 150, 150, 25);
add(tfPercentage);

// Calculate button
calculate = new JButton("Calculate");
calculate.setBounds(50, 200, 100, 25);
calculate.addActionListener(this);
add(calculate);

// Apply button
apply = new JButton("Apply");
apply.setBounds(170, 200, 100, 25);
apply.addActionListener(this);
add(apply);

// Back button
back = new JButton("Back");
back.setBounds(290, 200, 100, 25);
back.addActionListener(this);
add(back);

// Result label to show calculated values
resultLabel = new JLabel("");
resultLabel.setBounds(50, 250, 400,
25); add(resultLabel);

setVisible(true);
}

public void actionPerformed(ActionEvent ae) {
    String selectedEmpId =
    employeeId.getSelectedItemAt(); String type =
    cType.getSelectedItemAt();

    if (ae.getSource() == calculate) {

```



```

try {
    double percentage = Double.parseDouble(tfPercentage.getText());

    // Fetch current salary from the database
    Conn c = new Conn();
    ResultSet rs = c.s.executeQuery("SELECT basicSalary FROM employee WHERE empID
= '" + selectedEmpId + "'");

    if (rs.next()) {
        double basicSalary = rs.getDouble("basicSalary");

        // Calculate bonus or increment
        double amount = basicSalary * (percentage / 100);
        double newSalary = type.equals("Bonus") ? basicSalary + amount : basicSalary * (1 +
percentage / 100);

        resultLabel.setText(type + ": " + amount + ", New Salary: " + newSalary);
    }
} catch (Exception e)
{
    e.printStackTrace();
}
} else if (ae.getSource() == apply) {
    try {
        double percentage =
        Double.parseDouble(tfPercentage.getText()); Conn c = new
        Conn();
        ResultSet rs = c.s.executeQuery("SELECT basicSalary FROM employee WHERE empID
= '" + selectedEmpId + "'");

        if (rs.next()) {
            double basicSalary = rs.getDouble("basicSalary");
            double amount = basicSalary * (percentage / 100);
            double newSalary = type.equals("Bonus") ? basicSalary + amount : basicSalary * (1 +
percentage / 100);

            // Update the new salary and percentage in the database
            String updateQuery = "UPDATE employee SET basicSalary = " + newSalary +
            ", " + (type.equals("Bonus") ? "bonusPercent" : "incrementPercent") + " = "
+ percentage +
            " WHERE empID = '" + selectedEmpId + "'";
            c.s.executeUpdate(updateQuery);

            resultLabel.setText("New Salary Applied: " + newSalary);
            JOptionPane.showMessageDialog(null, type + " Applied Successfully");
        }
    } catch (Exception e)
    {
        e.printStackTrace();
    }
} else if (ae.getSource() == back) {

```

```
        setVisible(false)
        ; new Home();
    }
}

public static void main(String[] args) {
    new Payroll();
}
```

8.ATTENDANCE PAGE

```
package
```

```
employee.payroll.management; import
```

```
javax.swing.*;
```

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
import java.sql.*;
```

```
public class Attendance extends JFrame implements ActionListener {
```

```
    JComboBox<String> empIdComboBox;
```

```
    JButton searchButton;
```

```
    JTextArea detailsArea;
```

```
    Attendance() {
```

```
        setLayout(null)
```

```
    ;
```

```
        // Heading
```

```
        JLabel heading = new JLabel("Employee Attendance");
```

```
        heading.setBounds(320, 30, 500, 50);
```

```
        heading.setFont(new Font("SAN_SERIF", Font.BOLD,  
25)); add(heading);
```

```
        // Employee ID combo box
```

```
        JLabel labelEmpId = new JLabel("Select Employee
```

```
ID:"); labelEmpId.setBounds(50, 100, 150, 30);
```

```
labelEmpId.setFont(new Font("serif", Font.PLAIN, 20));
```

```
add(labelEmpId);
```

```
empIdComboBox = new JComboBox<>();
```

```
empIdComboBox.setBounds(220, 100, 150, 30);
```

```
add(empIdComboBox);
```

```

// Populate employee IDs from the
database try {
    Conn c = new Conn();
    ResultSet rs = c.s.executeQuery("SELECT empID FROM
employee"); while (rs.next()) {
        empIdComboBox.addItem(rs.getString("empID"));
    }
} catch (SQLException e)
    { e.printStackTrace();
}

// Search button
searchButton = new JButton("Search");
searchButton.setBounds(220, 150, 150, 30);
searchButton.addActionListener(this);
add(searchButton);

// TextArea to display employee details
detailsArea = new JTextArea();
detailsArea.setBounds(50, 200, 600, 300);
detailsArea.setFont(new Font("serif", Font.PLAIN, 18));
detailsArea.setEditable(false);
add(detailsArea);

setSize(700, 600);
setLocation(300,
50); setVisible(true);
}

@Override
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == searchButton) {
        String empId = (String) empIdComboBox.getSelectedItem();

        if (empId == null || empId.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please select a valid Employee ID.");
            return;
        }

        // Fetch employee attendance details from the database
        try {
            Conn c = new Conn();
            String query = "SELECT * FROM attendance WHERE empID = ?";

```

```

PreparedStatement pst = c.c.prepareStatement(query);
pst.setString(1, empId);
ResultSet rs = pst.executeQuery();

StringBuilder empDetails = new
StringBuilder(); boolean found = false;

while (rs.next()) {
    empDetails.append("Employee ID:
    ").append(rs.getString("empID")).append("\n"); empDetails.append("Date:
    ").append(rs.getString("date")).append("\n"); empDetails.append("Status:
    ").append(rs.getString("status")).append("\n");

    // Check if the leaveType is not null and append it
    String leaveType = rs.getString("leaveType");
    if (leaveType != null && !leaveType.isEmpty()) {
        empDetails.append("Leave Type: ").append(leaveType).append("\n");
    }

    String remarks = rs.getString("remarks");
    if (remarks != null && !remarks.isEmpty()) {
        empDetails.append("Remarks:
        ").append(remarks).append("\n");
    }

    empDetails.append("\n
    \n");
    found = true;
}

if (found) {
    detailsArea.setText(empDetails.toString());
} else {
    detailsArea.setText("No attendance details found for Employee ID: " + empId);
}
pst.close();
} catch (SQLException e)
{ e.printStackTrace();
JOptionPane.showMessageDialog(this, "Error fetching data: " + e.getMessage());
}
}
}
public static void main(String[] args) {
    new Attendance();
}
}


```

HOME PAGE



ADD EMPLOYEE PAGE

Add Employee Detail

Name	<input type="text" value="Haritha"/>	Father's Name	<input type="text" value="Mallaian"/>
Date of Birth	<input type="text" value="14 Jan 2006"/> 	Salary	<input type="text" value="50000"/>
Address	<input type="text" value="Ground floor, Swagath apts, Chrompet"/>		
Phone	<input type="text" value="9087388972"/>	Email	<input type="text" value="harithamallaian@gmail.com"/>
Education	<input type="text" value="BE"/> ▼	Designation	<input type="text" value="Manager"/>
Aadhar Number	<input type="text" value="996195034327"/>	Employee ID	<input type="text" value="AB435"/>

Submit

Cancel

—□×

Add Employee Detail

Name

Haritha

Father's Name

M

Date of Birth

14 Jan 2006

Salary

30000

Address

No.45, Swagath Apar

Chennai-600044

Phone

9087388972

harithamallaian@gmail.com

Education

BE

Intern

Aadhar Number

9961 9503 4327

Employee ID

AB453

Message

×

Employee added successfully!

OK

Submit

Cancel

UPDATE EMPLOYEE PAGE

Update Employee Detail

Employee ID

AB435

Search

Name

Father's Name

Date of Birth

Address

Phone

Salary

BBA

Designation

Aadhar Number

Message

×

No record found.

OK

Back

Select Employee ID:

BC123

Select Type:

Increment

Enter Percentage:

10

Calculate

Apply

Back

VIEW EMPLOYEE PAGE

Search by Employee ID:

Ab134


Search

Reset

Back

name	fname	dob	address	phone	email	education	designation	aadhar	emplID	bonusPerce...	incrementP
Abc	D	15 Oct 2000	Salem	9876347987	Abc@gmail...	BTECH	HR	4567 9876 3...	Ab134	0.0	0.0
Haritha	M	14 Jan 2006	No.45, Swa...	964575664	harithamalai...	BE	Intern	9961 9503 4...	AB453	0.0	0.0
abc	e	7 Nov 2024	Enathur	9868872355	gfdadgtd	BBA	HR	6879624287...	AS235	0.0	0.0
Gayatri	V R	7 Jan 2006	Kanyakumari	7896757443	gayatrivr@g...	BE	Manager	6879 3456 7...	BC123	0.0	0.0
Harini	P	22 Oct 2005	Central,Che...	8907556789	harinip@gm...	BE	Manager	9978 7785 3...	BC234	0.0	0.0
Harshita	M A	30 Aug 2005	Kolathur, Ch...	6473547573	harshitama...	BE	Manager	6785 3446 9...	BC245	0.0	0.0
Jyoshna	T	22 Feb 2006	Arakkonam,...	8769567845	joshnat@g...	BE	HR	9873 5678 8...	HR345	0.0	0.0
XYZ	A	15 Jan 1990	Radha Naga...	9088898753	xya@gmail...	B.COM	HR	9876543456...	XY123	0.0	0.0

PAYROLL PAGE

 Calculate Bonus or Increment

—□×

Select Employee ID:

BC123

Select Type:

Increment

Enter Percentage:

10

Calculate

Apply

Back

The development of the Employee Payroll Management System represents a significant advancement in streamlining payroll processes and enhancing the management of employee-related information. By leveraging modern technologies, the system achieves its primary objective of automating payroll calculations, improving data accuracy, and optimizing administrative efficiency, thus providing a seamless experience for both employees and administrators.

Utilizing Java and MySQL establishes a secure and efficient foundation for robust server-side processing and reliable database management. JavaFX or JSP and Servlets contribute to crafting a user-friendly and responsive interface, while the integration of reporting tools like JasperReports enhances the system's utility by enabling the generation of comprehensive reports.

With well-defined functional requirements, the system facilitates critical operations such as employee registration, attendance tracking, salary calculations, and payroll generation. Furthermore, the incorporation of non-functional requirements ensures high performance, security, scalability, reliability, and maintainability, guaranteeing uninterrupted service and user satisfaction.

In summary, the Employee Payroll Management System effectively addresses the challenges of payroll management by providing an integrated and adaptable solution. It not only meets operational needs but also sets a benchmark for efficiency and accuracy in managing employee data and payroll. The system's design and implementation offer a scalable and future-proof solution, ensuring consistent performance and contributing to streamlined organizational processes.

- [1] Soni, R., & Agarwal, A. (2021). *Payroll Management System: A Study*. Journal of Computer Science and Applications, 9(3), 50-60. Retrieved from https://journals.sonijournals.com/Payroll_Management_Study
- [2] Singh, M., & Gupta, R. (2019). *Design and Development of an Automated Payroll System Using MySQL and Java*. International Journal of Computer Science, 8(4), 77-84. Retrieved from https://www.ijcsitjournal.com/Automated_Payroll_System
- [3] https://www.businessjournal.com/payroll_optimization