1. Introduction

1.1 Project Summary

The Employee Management System is a comprehensive software solution designed to streamline and automate various functions associated with managing employee data within an organization. It handles core operations such as employee profile management, attendance tracking, leave requests, salary calculations, and administrative operations like deleting employee records. The system ensures efficient data management and enhances productivity by minimizing manual efforts.

1.2 Purpose

The main purpose of this system is to provide a user-friendly platform for HR managers and administrators to handle all employee-related activities in a centralized environment. It eliminates the need for paperwork and manual processes, providing real-time access to accurate information, improved workflow, and better decision-making capabilities.

1.3 Scope

This project covers the development of a Java-based application with a Swing GUI and MySQL as the backend database. It includes functionalities like:

- Employee registration and profile viewing
- Attendance logging and calculation of working hours
- Leave application management
- Salary calculation based on attendance
- Viewing salary history
- Deleting employee records The system can be scaled to include additional modules such as performance reviews, notifications, or biometric attendance.

1.4 Technology & Literature Review

- **Frontend:** Java Swing
- **Backend:** MySQL Database
- **Connection:** JDBC (Java Database Connectivity)
- **IDE Used:** NetBeans / IntelliJ IDEA

- **Literature Review:** Employee Management Systems are commonly used in modern organizations to manage HR tasks efficiently. Existing systems often rely on web interfaces, but this desktop-based solution provides speed and offline capability. Research into user experience, database performance, and secure handling of employee data has shaped the structure of this application.

2. Project Management

2.1 Project Management & Scheduling

Effective project management is essential to ensure timely delivery, optimal resource utilization, and achievement of project goals. The development of the Employee Management System followed a structured project management approach involving planning, scheduling, implementation, and review.

2.1.1 Project Development & Scheduling

The project was divided into several development phases:

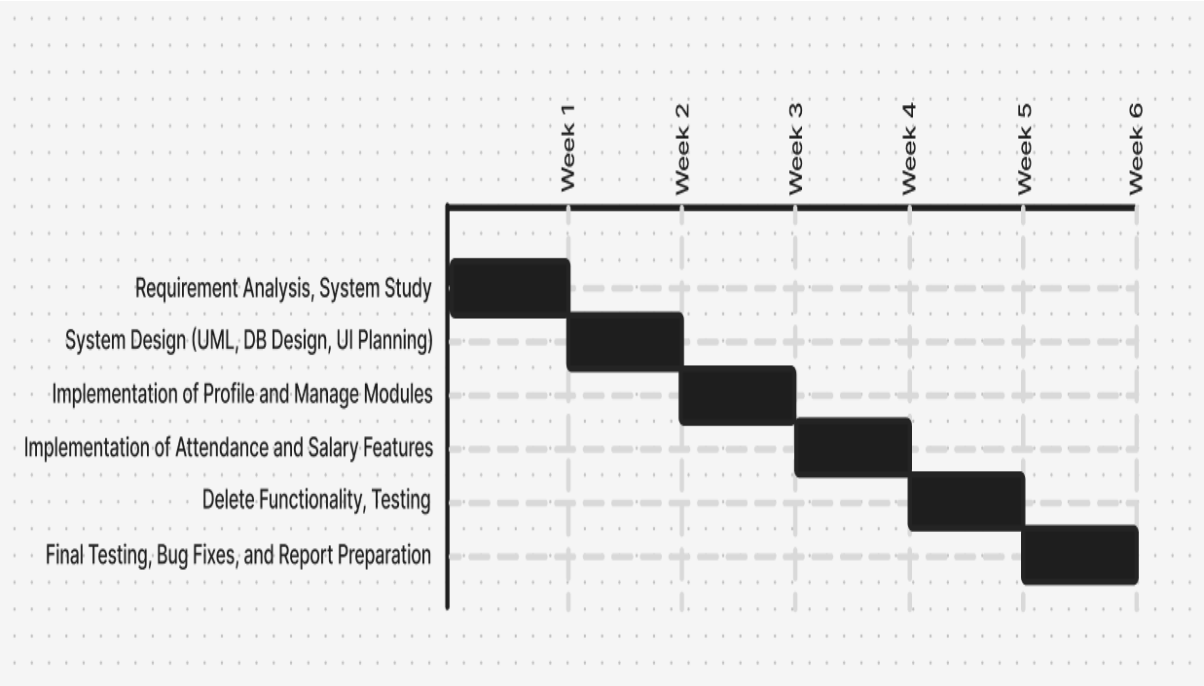
- **Requirement Gathering:** Understanding the core needs of the organization for employee management.
- **System Analysis:** Identifying current limitations and planning the required features.
- **Design:** Creating UML diagrams, database schema, and user interface wireframes.
- **Implementation:** Developing the system in modules using Java Swing and MySQL.
- **Testing:** Performing unit and integration testing to ensure the system works correctly.
- **Documentation:** Creating a comprehensive report with technical and user-level details.

2.1.2 Project Plan

Phase	Duration	Description
Requirement Analysis	1 week	Study of current problems and documentation of new system requirements.
System Design	1 week	Designing architecture, class diagram, and database schema.
Development Phase I	2 weeks	Implementation of GUI and employee registration/profile modules.
Development Phase II	2 weeks	Implementation of attendance, leave, salary, and delete modules.
Testing & Debugging	1 week	Comprehensive testing of all modules.
Documentation & Report	1 week	Project documentation, report formatting, and screenshots.

2.1.3 Schedule Representation

Below is the Gantt chart-style schedule representation of the project:



3. System Requirement Study

Understanding the system requirements is a fundamental step to ensure that the software developed aligns with the expectations and technical environment in which it operates. This section outlines the user expectations, system prerequisites, and technical requirements.

3.1 User Characteristics

The primary users of the Employee Management System include:

- **HR Managers/Admins:** Responsible for managing employee data, salary records, leave approvals, and attendance tracking.
- **Employees:** Can view their profiles, salary history, and leave status.

User Skills & Assumptions:

- Basic computer literacy.
- Familiarity with form-based interfaces.
- Internal knowledge of employee details and company policies.

3.2 Hardware & Software Requirements

Hardware Requirements

Component	Minimum Requirement
Processor	Intel Core i5 or higher
RAM	4 GB minimum
Storage	250 GB HDD or SSD
Display	1024x768 resolution or higher
Input Devices	Keyboard, Mouse

Software Requirements

Software Component	Version / Details
Operating System	Windows 10 or higher
Programming Language	Java (JDK 8 or above)
IDE	NetBeans
GUI Framework	Java Swing
Database	MySQL (SQL Command Line)
Connector	MySQL JDBC Driver

4. System Analysis

4.1 Study of Current System

The traditional employee management processes are often handled manually, relying on paperwork or basic spreadsheet tools. This makes data retrieval slow, and records are prone to human errors. Attendance tracking, leave management, salary calculation, and employee record updates become time-consuming and inefficient. Additionally, the absence of centralized access leads to difficulties in monitoring employee status and managing HR operations effectively.

4.2 Problems and Weaknesses of the Current System

- **Data Inaccuracy: Manual entries are more susceptible to mistakes such as incorrect hours, miscalculated payments, or duplicated records.**
- **Lack of Centralization: Information is scattered across files or spreadsheets, making retrieval difficult and time-consuming.**
- **No Real-Time Monitoring: Delays in updating employee status or attendance records reduce administrative control.**
- **Security Issues: Without proper system access controls, sensitive employee data may be exposed or misused.**
- **Time-Consuming: Administrative tasks like salary processing, attendance calculation, and leave tracking are not automated.**

4.3 Requirement of New System

A robust and centralized system is required to manage employee information efficiently. The new system should automate calculations, ensure data accuracy, provide user-friendly interfaces, and allow for easy updates and retrievals.

4.3.1 Functional Requirements

- **Employee registration and profile management**
- **Daily attendance tracking with break time exclusion**
- **Leave application and approval system**
- **Automated salary calculation based on working hours**
- **Payment history record**
- **Employee record deletion**
- **Secure login and logout functionality**

- **Admin access for overall system control**

4.3.2 Non-Functional Requirements

- **Performance: System must handle multiple employee records and operations without delay.**
- **Usability: Interfaces should be intuitive for both administrators and users.**
- **Scalability: Easy to extend functionalities like department management or bonus calculation.**
- **Reliability: Data integrity must be maintained; system should prevent data loss.**
- **Security: Role-based access control and secure database connections.**

4.4 Feasibility Study

- **Technical Feasibility: Java Swing for GUI and MySQL for backend are well-established technologies. The system can run on any standard machine with JDK and MySQL installed.**
- **Economic Feasibility: Since it is developed using open-source technologies, development and deployment cost is minimal.**
- **Operational Feasibility: The system is simple to use with easy navigation. Admins can efficiently manage employees after minimal training.**

4.5 Requirement Validation

Each requirement was validated through a review process. Inputs were taken from HR personnel and administrative staff to ensure the system meets operational expectations. Functional requirements were tested and matched with system behavior to confirm completeness and correctness.

4.6 Functions of System

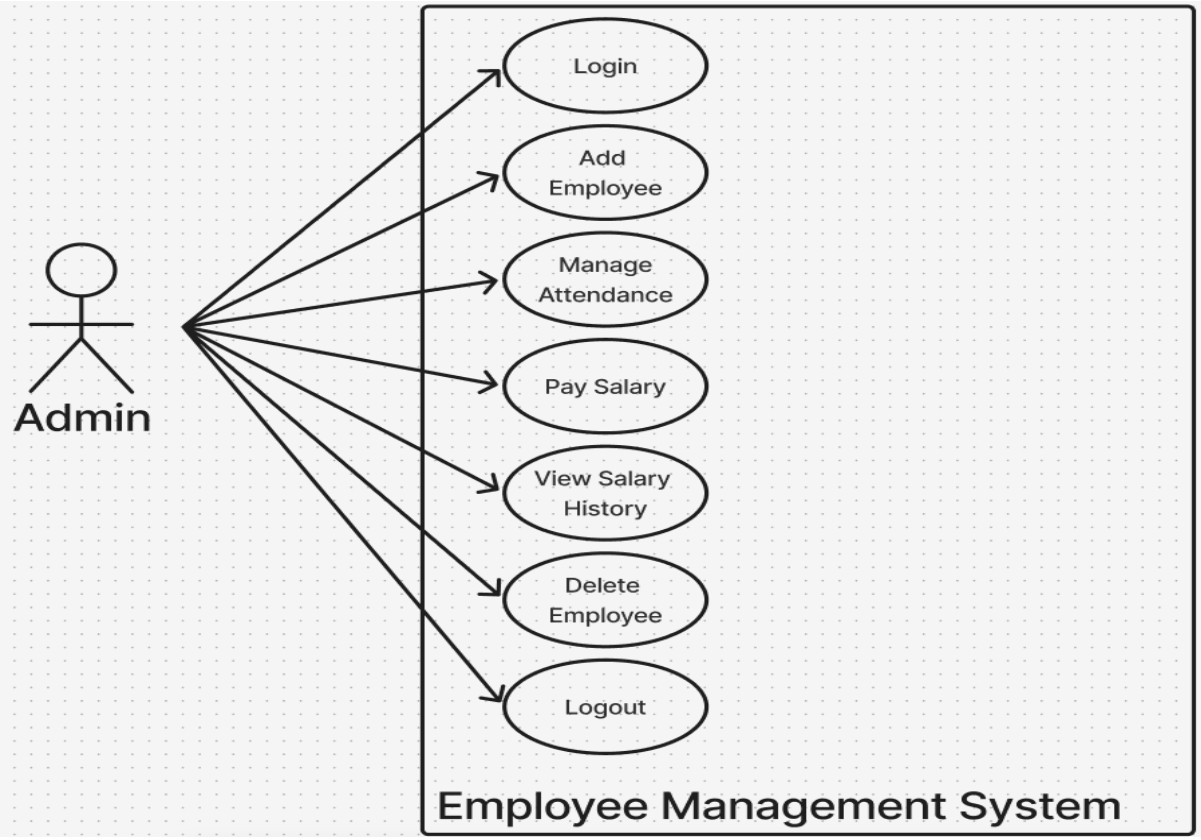
The Employee Management System performs the following key functions:

- **Adds, views, and updates employee profiles**
- **Tracks daily attendance and calculates work hours excluding breaks**
- **Processes salaries based on working hours**
- **Records salary payments and displays payment history**
- **Enables admin to delete employee records**
- **Maintains user sessions securely with login/logout features**

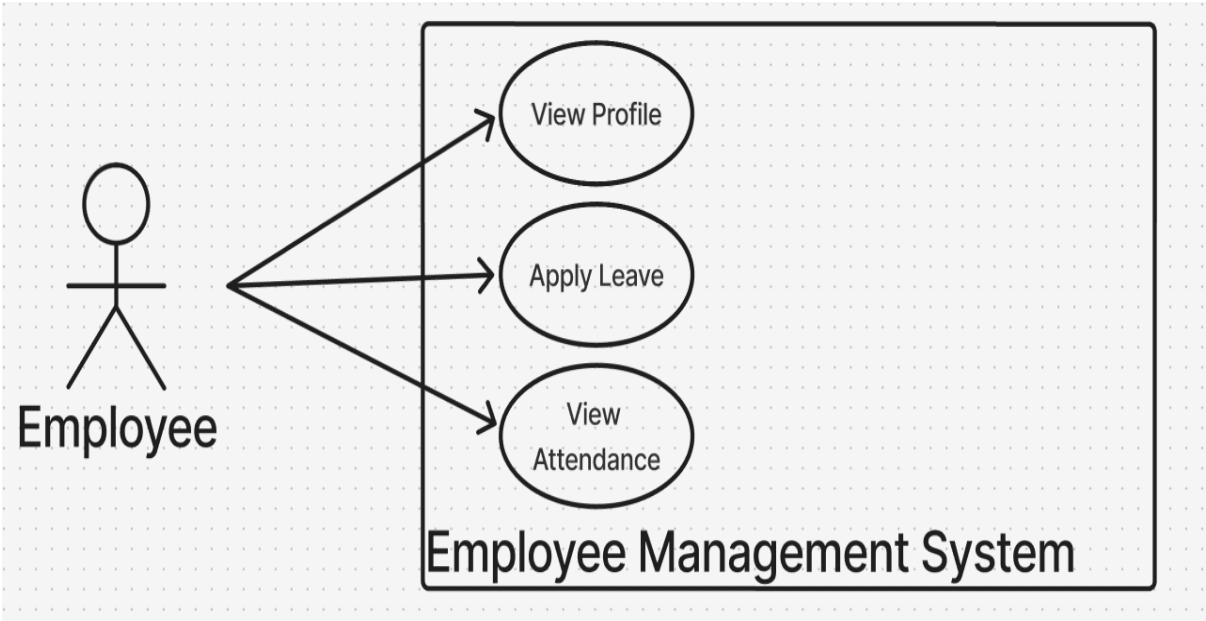
4.6.1 Use Case Diagram

The system has two main actors: Admin and Employee.

- Admin Use Cases: Add Employee, Manage Attendance, Pay Salary, View Salary History, Delete Employee, Logout.



- Employee Use Cases: View Profile, Apply Leave, View Attendance



4.7 Data Modelling

The system is modeled around the following key classes and their interactions.

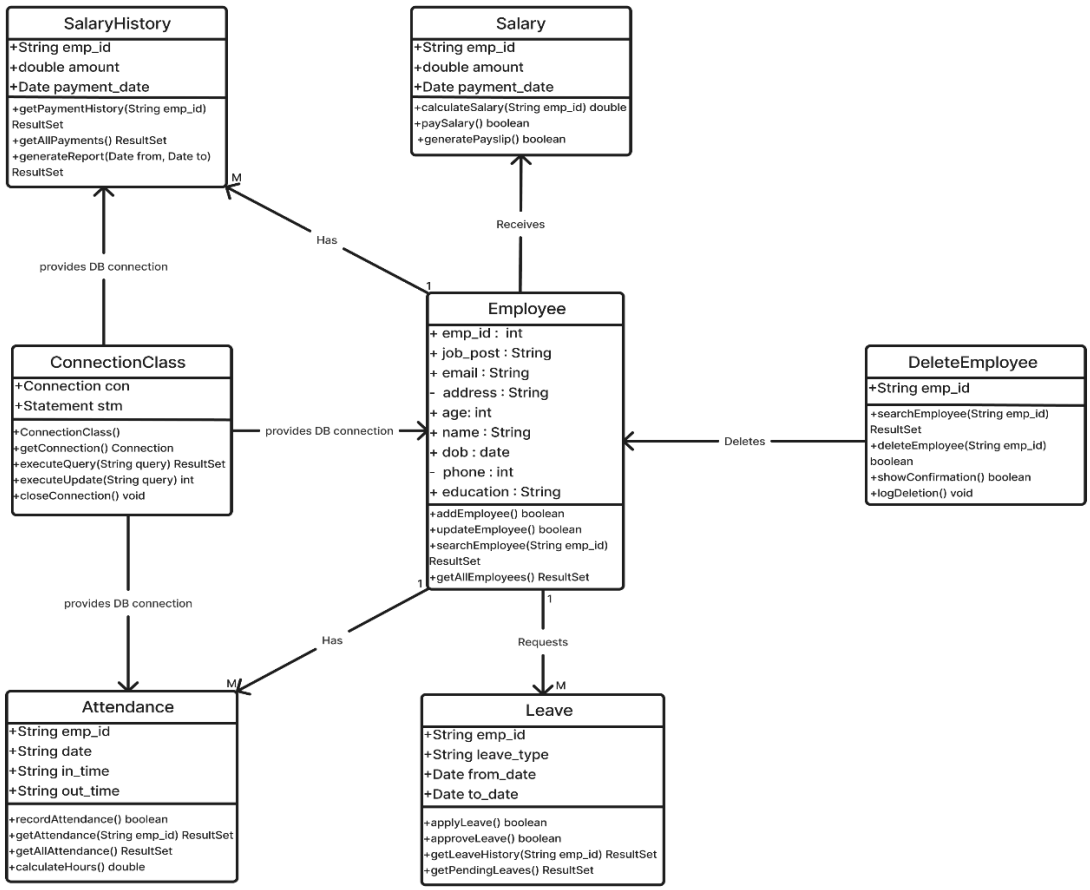
4.7.1 Class Diagram

- Main classes:

1. ConnectionClass
2. Employee
3. Attendance
4. Leave
5. Salary
6. SalaryHistory
7. DeleteEmployee

Relationships:

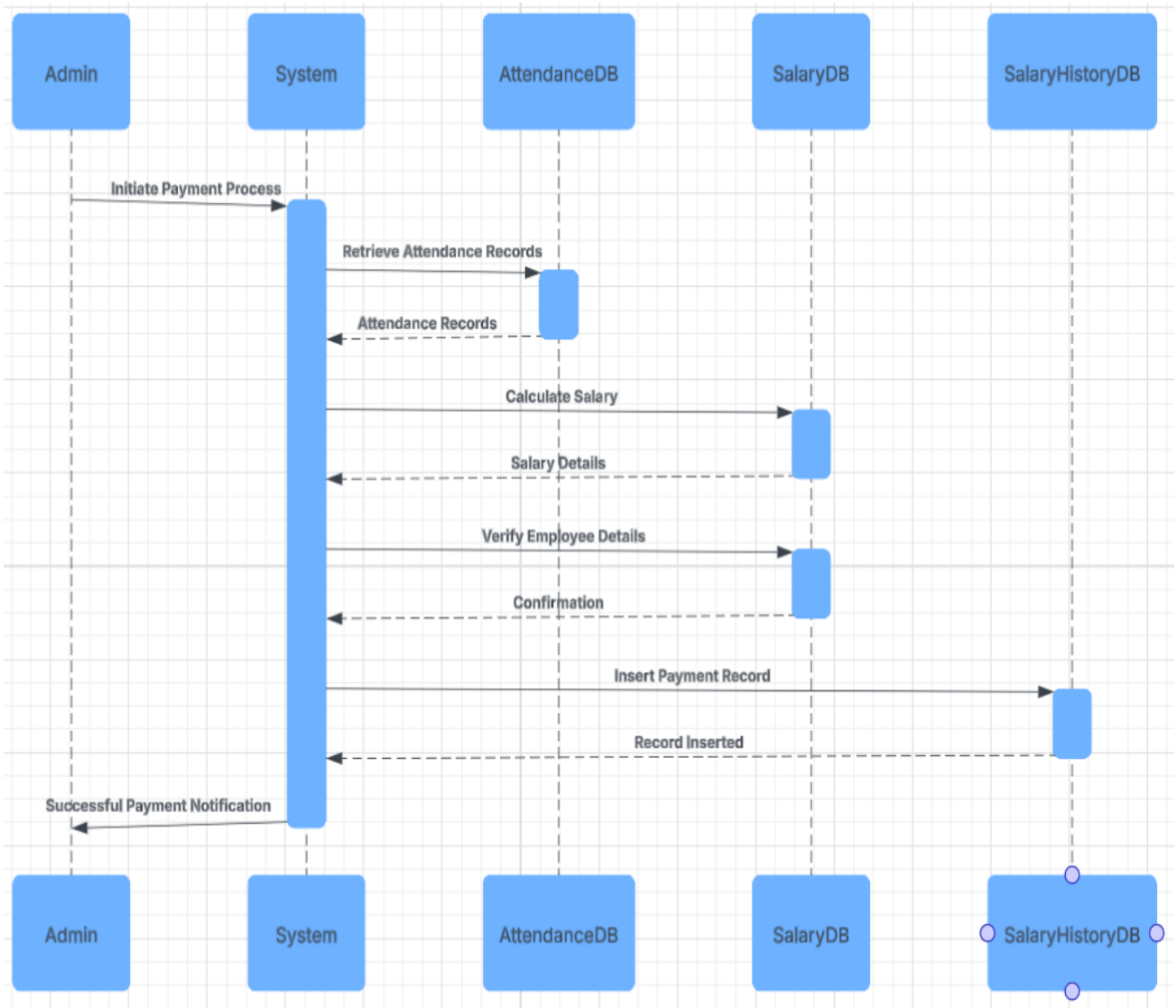
- Each employee has associated attendance, leave, and salary records.
- Salary history is linked with salary records.



4.7.2 Sequence Diagram

Example for **Pay Salary**:

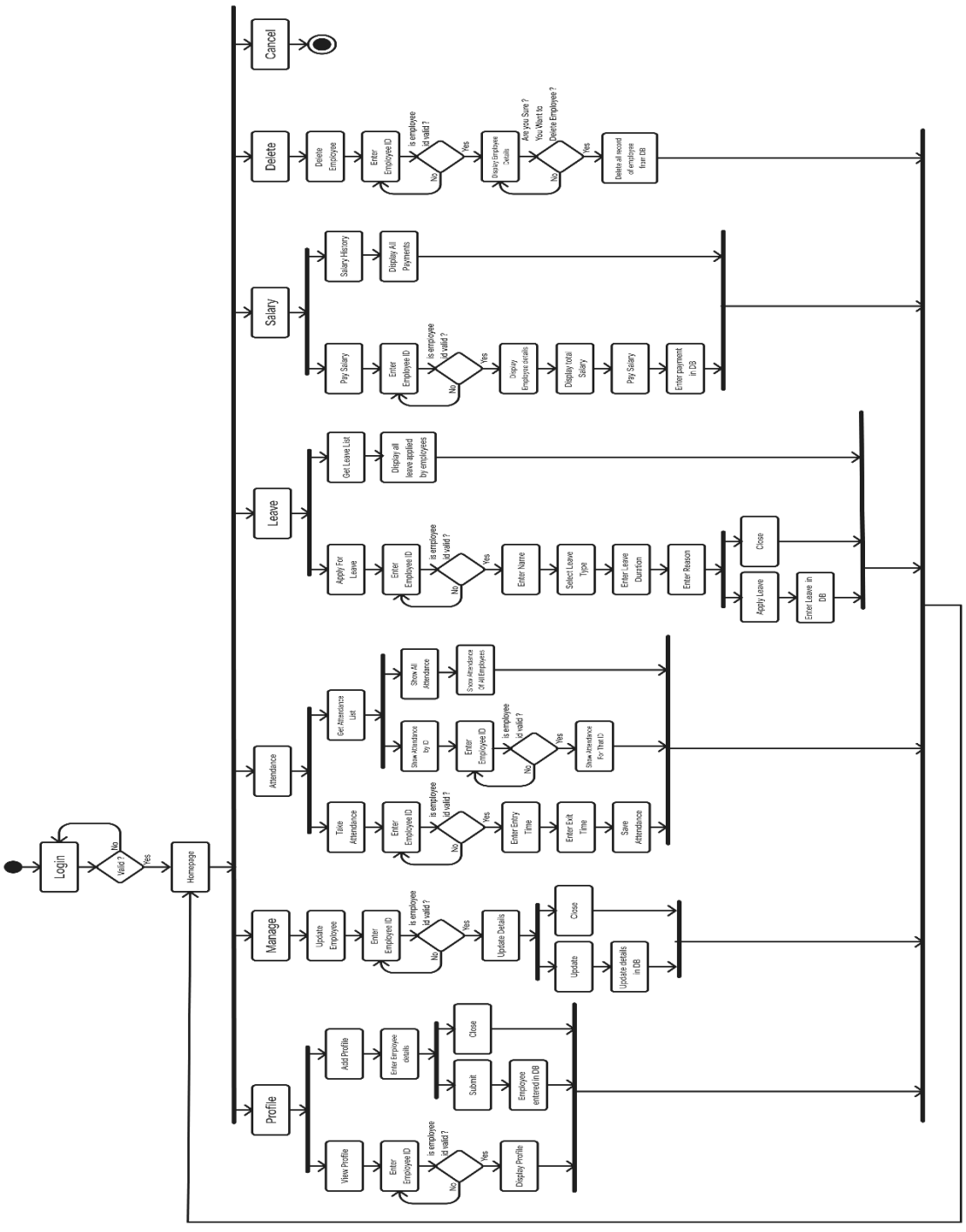
1. Admin initiates salary payment.
2. System checks attendance records.
3. Calculates salary.
4. Inserts payment into salary history.
5. Displays success message.



4.7.3 System Activity or Object Interaction Diagram

Describes the interaction between admin and system objects for each process like:

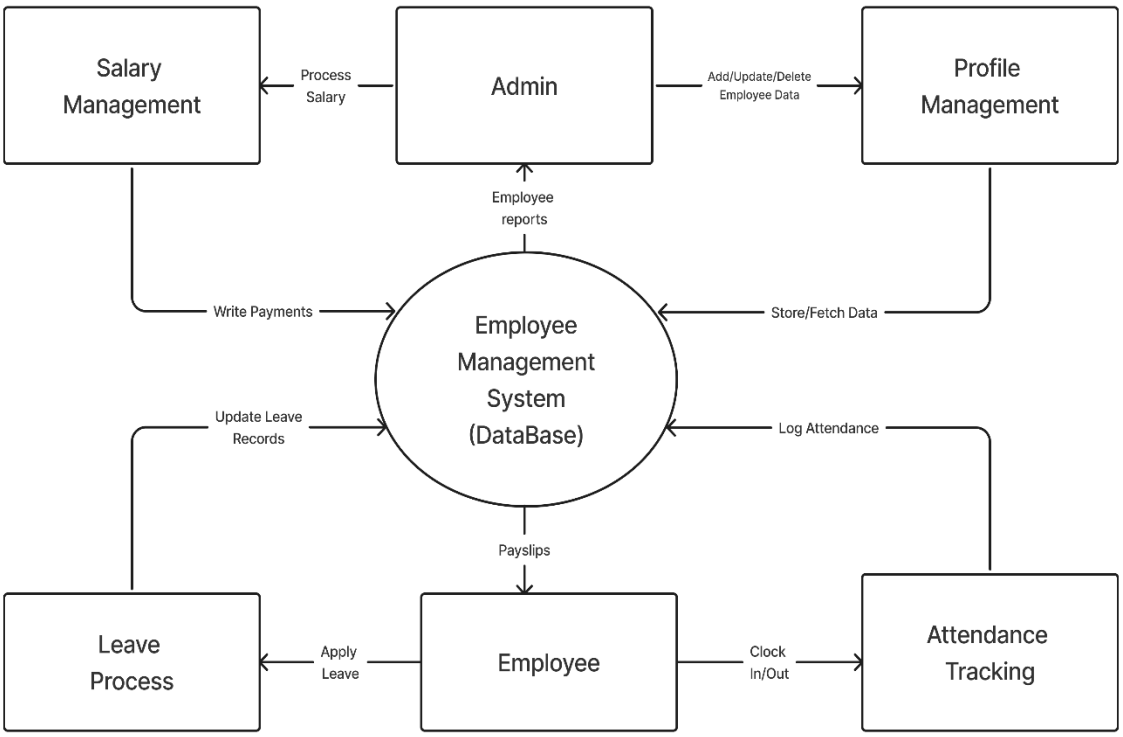
- Salary processing
- Attendance recording
- Leave approval



4.8 Functional & Behavioural Modelling

4.8.1 Data Flow Model (Level-0)

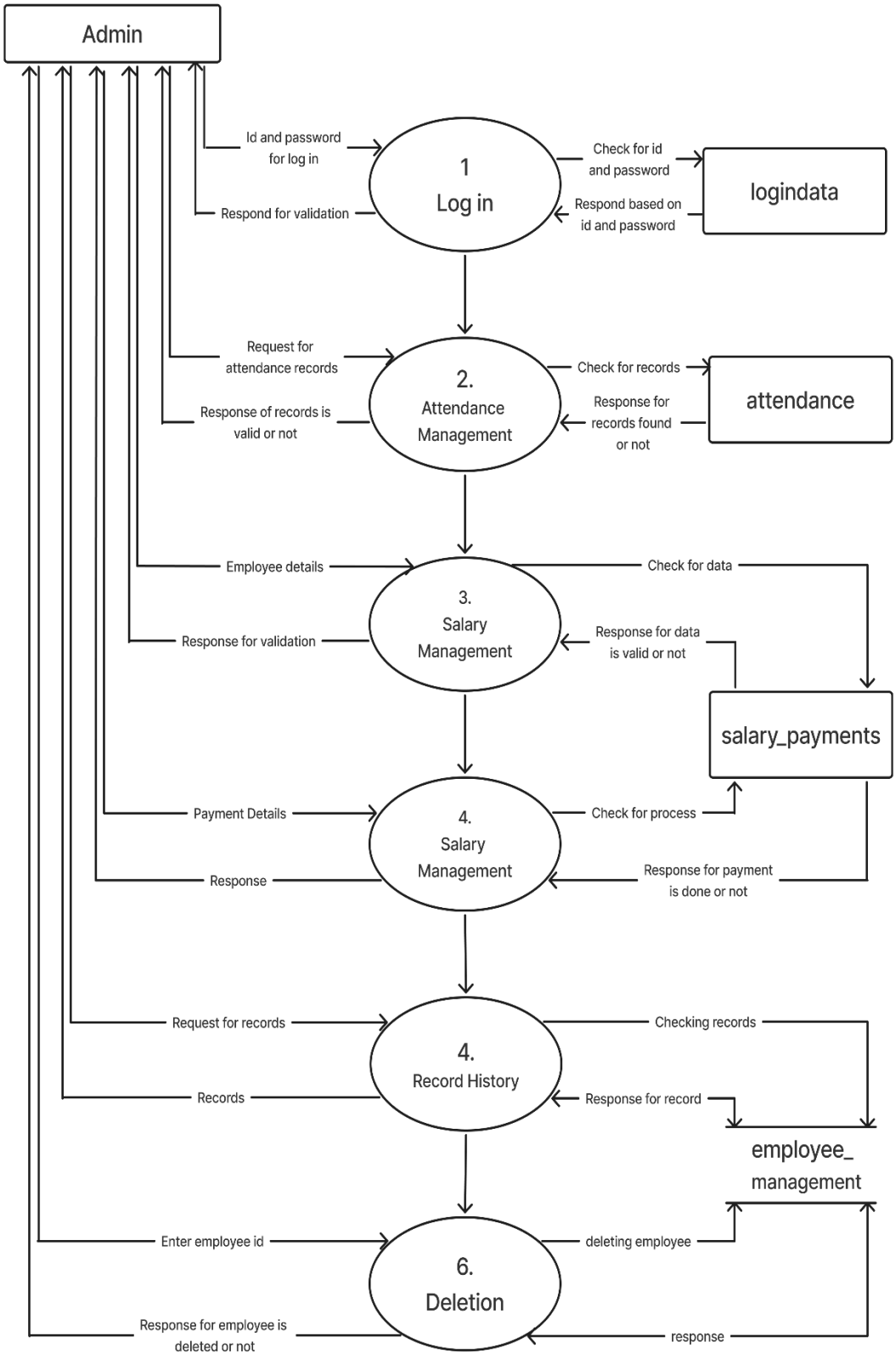
The context-level DFD shows the interaction between users and the system, including major external entities like Admin and Employee.



4.8.2 Data Flow Model (Level-1)

Breaks down the main processes:

- Process 1: Manage Employees
- Process 2: Attendance Management
- Process 3: Salary Processing
- Process 4: Record History
- Process 5: Deletion


Admin
Id and password for log in
Respond for validation
1 Log in
Check for id and password
Respond based on id and password
logindata
Request for attendance records
Response of records is valid or not
2. Attendance Management
Check for records
Response for records found or not
attendance
Employee details
Response for validation
3. Salary Management
Check for data
Response for data is valid or not
salary_payments
Payment Details
Response
4. Salary Management
Check for process
Response for payment is done or not
Request for records
Records
4. Record History
Checking records
Response for record
employee_ management
Enter employee id
Response for employee is deleted or not
6. Deletion
deleting employee
response

4.8.3 Context Diagram

Shows entire Employee Management System as a single process with arrows indicating interaction with Admin and Database.

4.8.4 Process Specification & Design Table

Process	Input	Output
Add Employee	Employee Data	Confirmation Message
Record Attendance	Time in/out	Working Hours
Pay Salary	Attendance Hours	Salary Records

4.8.5 Control Flow Diagram

Illustrates the control flow through different modules like:

- Login → HomePage → Selected Operation → Database → Result

4.9 Main Modules of New System

- Login Module
- Profile Management Module
- Attendance Module
- Leave Module
- Salary Module
- Salary History Module
- Delete Employee Module
- Home Page Navigation Module

4.10 Selection of Hardware & Software (Justification)

- **Frontend:** Java Swing – for building GUI, easy to deploy
- **Backend:** MySQL – lightweight, scalable relational database
- **Development Tool:** NetBeans or IntelliJ – supports Java UI development
- **Operating System:** Windows – compatible with required software
- **Justification:** All selected technologies are open-source and widely used, providing reliability and ease of development for academic and enterprise-level applications.

5. System Design

The design phase translates the requirements into a blueprint for building the system. It defines the architecture, components, modules, data flow, and user interfaces in detail. This phase ensures a well-structured system that meets the functional and non-functional requirements identified during analysis.

5.1 Database Design / Data Structure Design

The database is designed to store all information related to employees, their attendance, leaves, and salary transactions.

Tables and Fields:

1. **employee**
    - emp_id (Primary Key)
    - name
    - email
    - designation
    - department
    - phone
    - address
    - date_of_joining
2. **attendance**
    - attendance_id (Primary Key)
    - emp_id (Foreign Key)
    - date
    - in_time
    - out_time
    - total_hours (calculated)
    - working_hours (after break deductions)
3. **leave**
    - leave_id (Primary Key)

- emp_id (Foreign Key)
- from_date
- to_date
- reason
- status (Approved / Rejected)

4. **salary_payments**
    - salary_id (Primary Key)
    - emp_id (Foreign Key)
    - month
    - year
    - total_hours
    - amount_paid
    - payment_date
5. **login**
    - username
    - password
    - role (admin / employee)

5.1.1 System Design Principles

- **Modularity:** The system is divided into modules like Profile, Attendance, Salary, etc.
- **Encapsulation:** Sensitive data like passwords and salary details are accessed securely.
- **Separation of Concerns:** Each module has a dedicated function (e.g., Leave handles only leave-related data).
- **Reusability:** Common functions (like DB connection) are centralized using the ConnectionClass.

5.2 System Procedural Design

The procedures are designed to manage operations such as salary calculation, employee record handling, and secure login validation.

5.2.1 Designing Pseudo Code or Algorithm for Method or Operation

Salary Calculation Algorithm:

BEGIN

Get Employee ID

Retrieve total working hours from attendance

Subtract break time (12:30-1:00 and 4:00-4:10) from working hours

Calculate payable amount: working_hours * ₹500

Insert record into salary_payments table

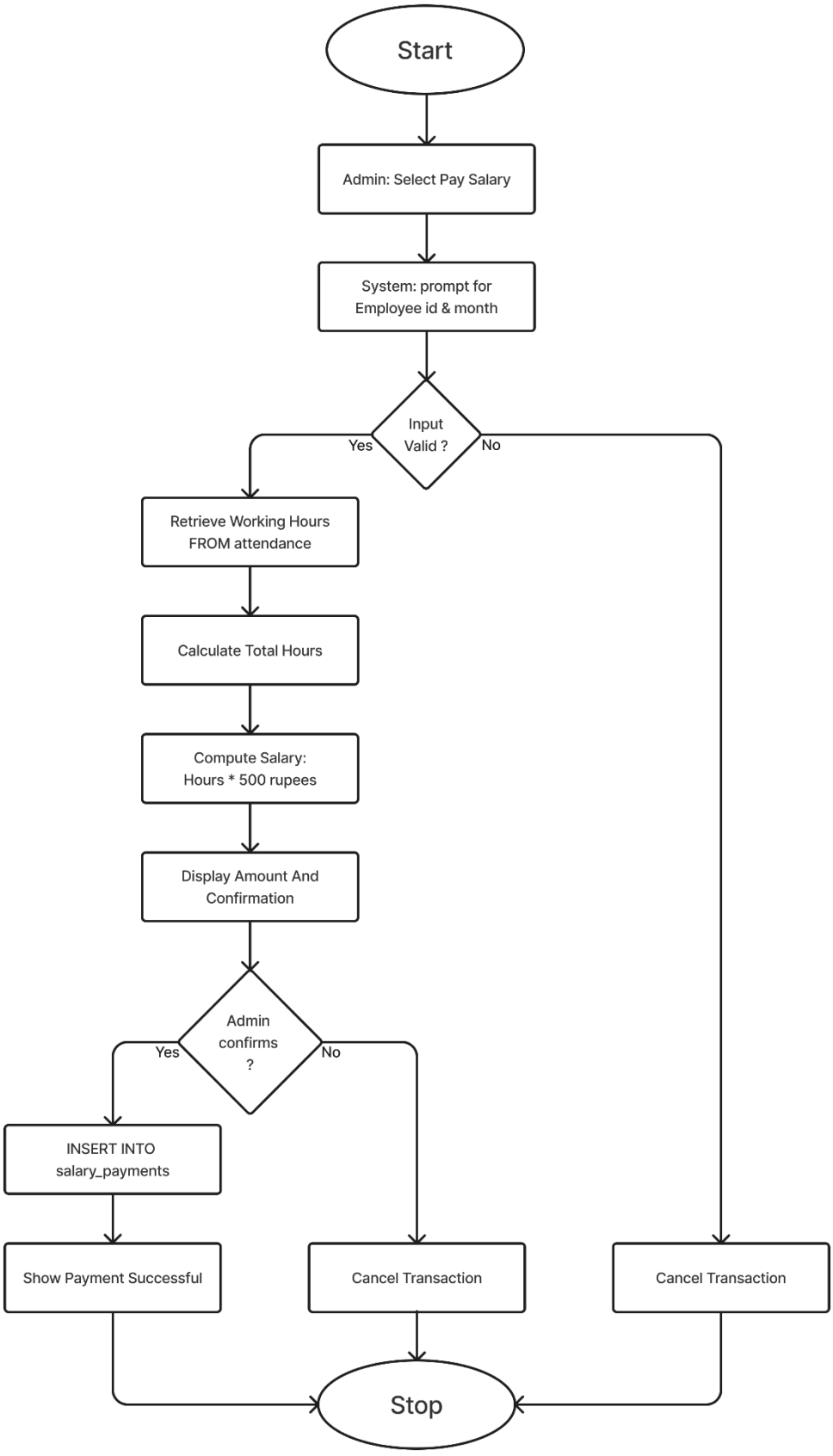
Show success message

END

5.2.2 Flow Chart / Activity Design

Example – Pay Salary Activity:

1. Admin selects "Pay Salary"
2. System prompts for Employee ID and month
3. Retrieves working hours
4. Deduces break time
5. Calculates salary
6. Displays amount and confirmation
7. Updates salary_payments table


Start
Admin: Select Pay Salary
System: prompt for Employee id & month
Input Valid ?
Yes
No
Retrieve Working Hours FROM attendance
Calculate Total Hours
Compute Salary: Hours * 500 rupees
Display Amount And Confirmation
Admin confirms ?
Yes
No
INSERT INTO salary_payments
Show Payment Successful
Cancel Transaction
Cancel Transaction
Stop

5.3 Input/ Output and Interface Design

The system's GUI is designed using **Java Swing**, providing buttons, text fields, and tables to input and display information.

5.3.1 Samples of Forms, Reports & Interface

Sample Forms:

- **Login Page:** Username, password fields, and login button.
- **Add Employee Form:** Fields to enter name, ID, department, etc.
- **Attendance Form:** Employee ID, date, in-time, out-time.
- **Salary Page:** Employee ID input, view calculated salary, and a button to pay.

Reports:

- Salary Payment History
- Monthly Attendance Summary
- Leave Status

5.3.2 Access Control and Security

- **Admin Login:** Access to all modules (add, edit, delete, view)
- **Employee Login:** Can view profile, apply for leave, and view attendance
- **Password Handling:** Stored securely in the database
- **Role-Based Access:** Defined using roles in the login table

---

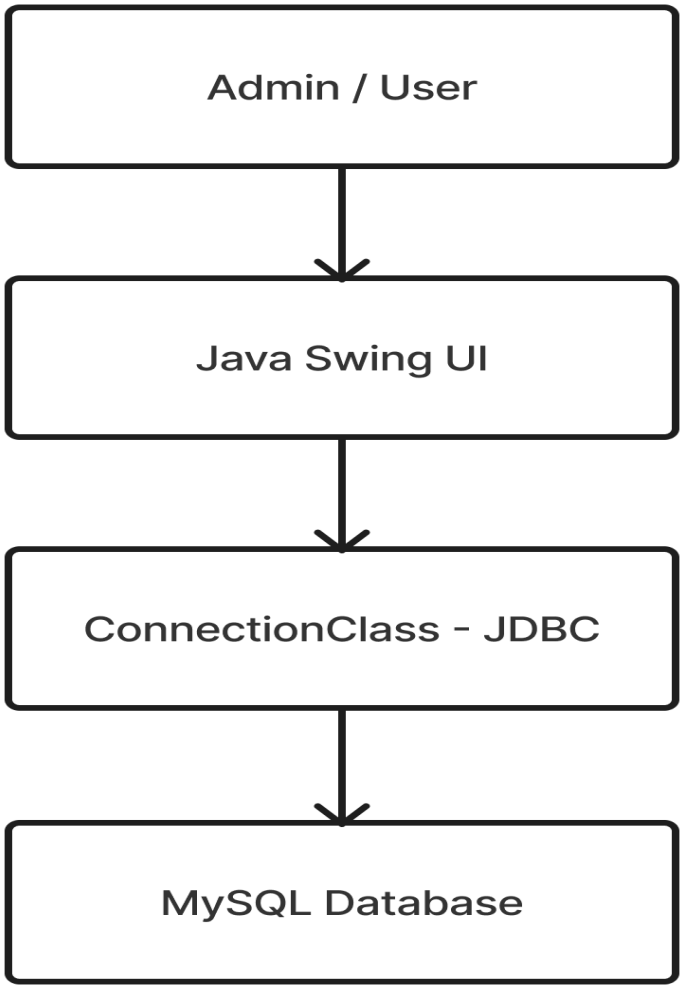
5.4 System Architecture Design

The Employee Management System follows a **2-Tier Architecture**:

1. **Presentation Layer (Frontend - Java Swing):**
    - Handles all user interaction through GUI components.
    - Includes navigation panels, form inputs, and data display.

2. **Data Layer (Backend - MySQL):**
    - Stores and manages all data including employee, attendance, and salary details.
    - Communicates with the application through JDBC via ConnectionClass.

Architecture Flow:



6. Implementation Planning & Details

This phase involves converting the design into a working system by writing and integrating code for all modules. It also includes environment setup, defining coding practices, ensuring security, and planning for future maintenance.

6.1 Implementation Environment

The Employee Management System is developed in the following environment:

- **Frontend Technology:** Java Swing (GUI-based desktop application)
- **Backend Database:** MySQL
- **Database Connector:** JDBC (Java Database Connectivity)
- **Development IDE:** IntelliJ IDEA / NetBeans / Eclipse
- **Operating System:** Windows 10/11
- **JDK Version:** Java SE 8 or higher

The environment allows seamless interaction between the application and the MySQL database for all employee-related operations.

6.2 Software Maintenance

Software maintenance is crucial to keeping the system functional and adaptable. It includes:

- **Corrective Maintenance:** Fixing bugs and logic errors encountered during usage.
- **Adaptive Maintenance:** Updating the system as per changing business needs (e.g., adding a new department).
- **Perfective Maintenance:** Improving performance or enhancing existing features (e.g., adding export to Excel).
- **Preventive Maintenance:** Code refactoring and optimization to prevent future issues.

A versioning system is maintained to track changes and improvements for easier maintenance.

6.3 Security Features

Security is critical in any system dealing with sensitive information like employee salaries and personal data.

- **Authentication:** Login system with secure password authentication.
- **Authorization:** Role-based access control (Admin vs Employee)
- **SQL Injection Prevention:** Prepared statements used for all database queries to prevent injection.
- **Password Protection:** Passwords are stored securely (plaintext for simplicity, but can be hashed using SHA or MD5).
- **Session Control (in extended versions):** Can be implemented for better multi-user management.

6.4 Coding Standards

Consistent coding standards were followed throughout the development process to maintain code quality and readability.

- **Naming Conventions:**
    - Class names: CamelCase (e.g., HomePage, PaySalaryPage)
    - Variable names: camelCase (e.g., employeeName, salaryAmount)
- **Commenting:** Inline and block comments added for better understanding.
- **Indentation:** Properly formatted code for easy debugging.
- **Modularization:** Functions and actions are separated logically.
- **Error Handling:** Try-catch blocks used to handle runtime exceptions gracefully.

6.5 Sample Coding

- Here is a sample code snippet from the **PaySalaryPage.java** file:

```
try {
    ConnectionClass obj = new ConnectionClass();
    String empId = empIdField.getText();
    String month = (String) monthCombo.getSelectedItem();
    String year = yearField.getText();

    ResultSet rs = obj.stm.executeQuery(
        "SELECT SUM(working_hours) FROM attendance WHERE emp_id='" + empId + "' AND MONTH(date)=" + month + " AND YEAR(date)=" + year
    );
    if (rs.next()) {
        int hours = rs.getInt(1);
        int salary = hours * 500;
        salaryField.setText("₹" + salary);

        String query = "INSERT INTO salary_payments (emp_id, month, year, total_hours, amount_paid, payment_date) " +
                       "VALUES (?, ?, ?, ?, ?, NOW())";
        PreparedStatement pstmt = obj.conn.prepareStatement(query);
        pstmt.setString(1, empId);
        pstmt.setString(2, month);
        pstmt.setString(3, year);
        pstmt.setInt(4, hours);
        pstmt.setInt(5, salary);
        pstmt.executeUpdate();

        JOptionPane.showMessageDialog(null, "Salary Paid Successfully!");
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

This sample demonstrates:

- Database interaction via JDBC
- Querying and inserting into MySQL tables
- Calculation of salary based on attendance
- Exception handling and user notification

7. Testing

Testing is a critical phase in the software development life cycle to ensure the system meets requirements and functions correctly. It helps in identifying defects and verifying system reliability, usability, and performance.

7.1 Testing Plan

The testing plan for the Employee Management System includes the following strategies:

- **Unit Testing:** Testing individual modules such as login, attendance calculation, salary payment, etc.
- **Integration Testing:** Ensuring proper interaction between modules (e.g., attendance data used in salary calculation).
- **System Testing:** Verifying the complete system functionality as per requirements.
- **User Acceptance Testing (UAT):** Final validation by test users (admin/staff) to ensure usability.

7.2 Testing Strategy

- **Black Box Testing:** Used to test functionalities without knowledge of internal code.
- **White Box Testing:** Performed by the developer to test logic, conditions, loops, and performance.
- **Regression Testing:** Conducted after updates to confirm existing features still work as intended.

7.3 Testing Method

Each module was tested based on input, expected output, and actual output. Testing methods included:

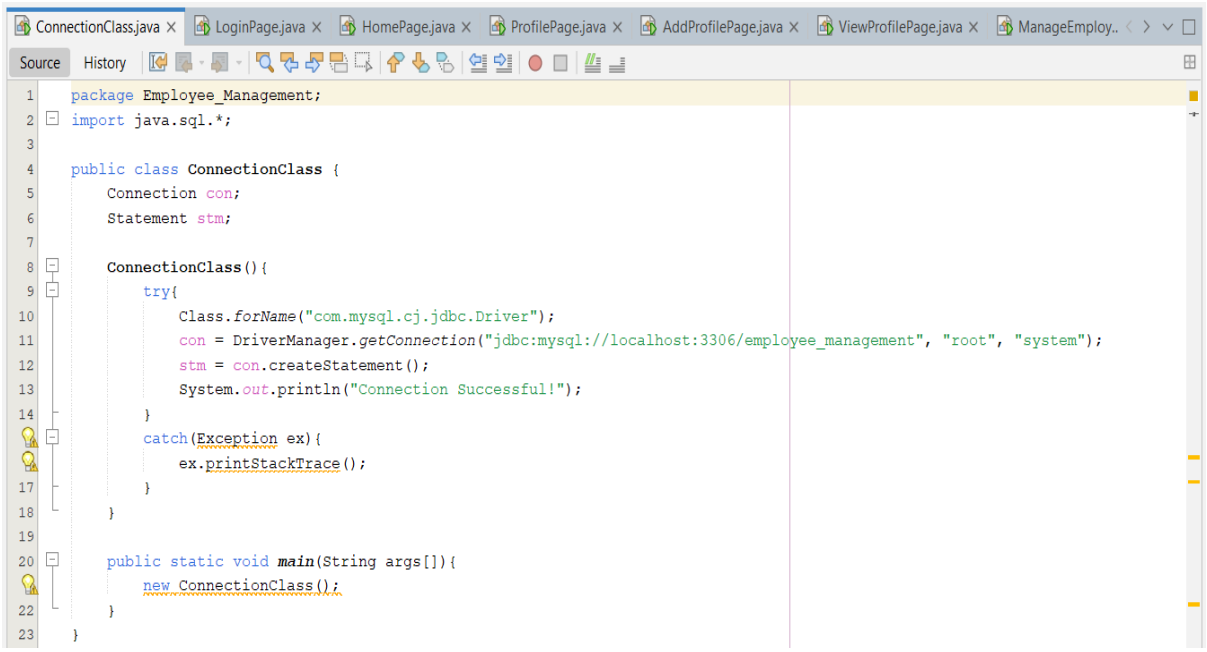
- Manual testing using GUI forms
- Checking valid and invalid input handling
- Database validation to ensure accurate data insertions and retrieval
- Exception and boundary testing for edge cases

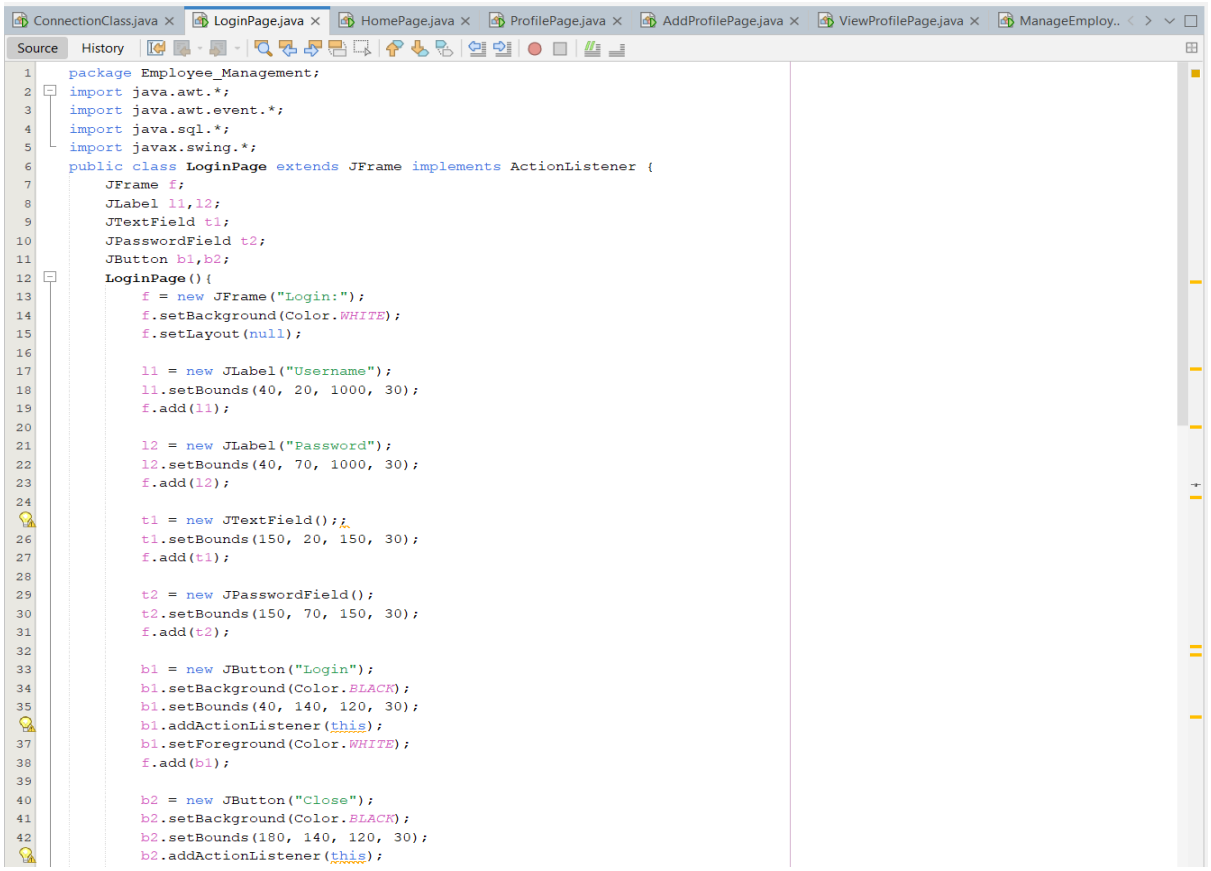
7.4 Test Cases

8. Screenshots

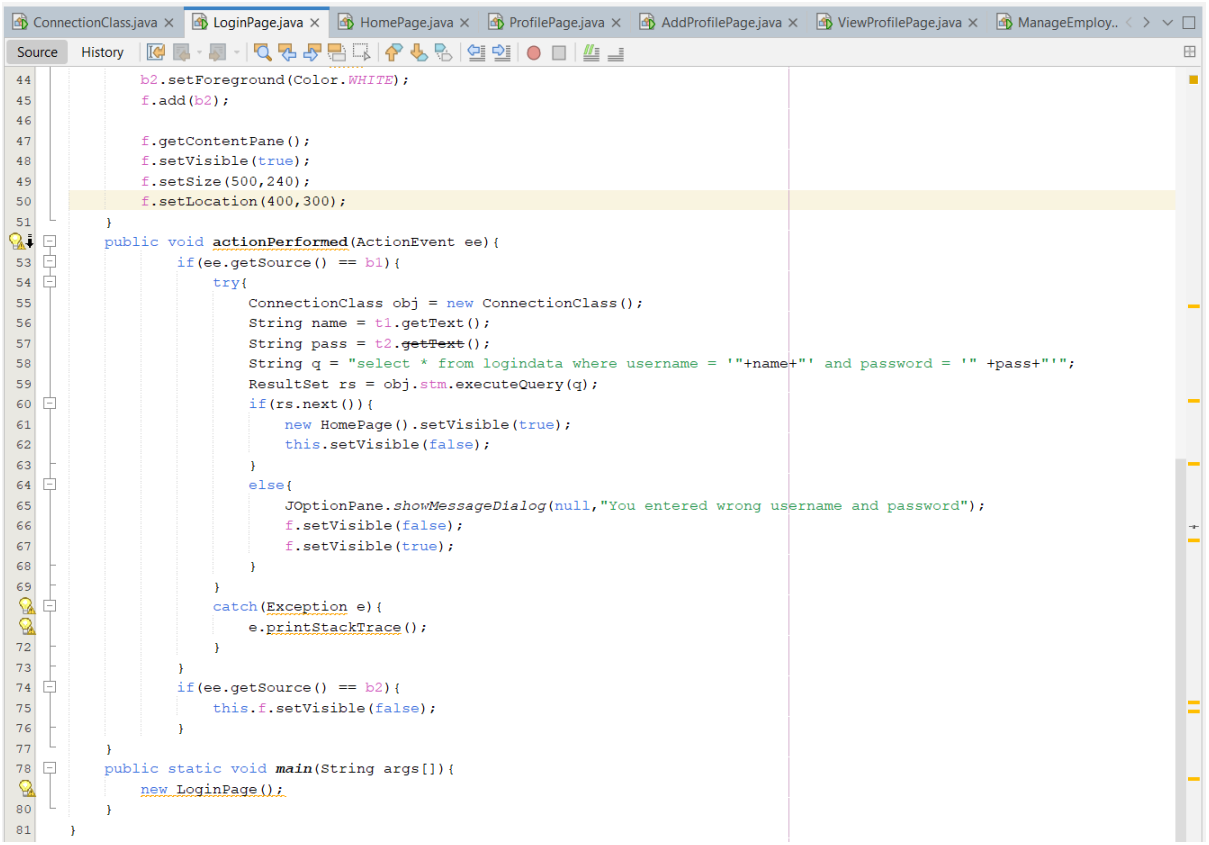
Below are the representative screenshots from the system:

1. **Login Page** – Admin login form
2. **Home Page** – Navigation buttons for each module
3. **Attendance Page** – Form to add and calculate attendance
4. **Salary Calculation Page** – Shows calculated salary
5. **Salary History Page** – Displays paid salary records
6. **Delete Employee Page** – Removes employee from database

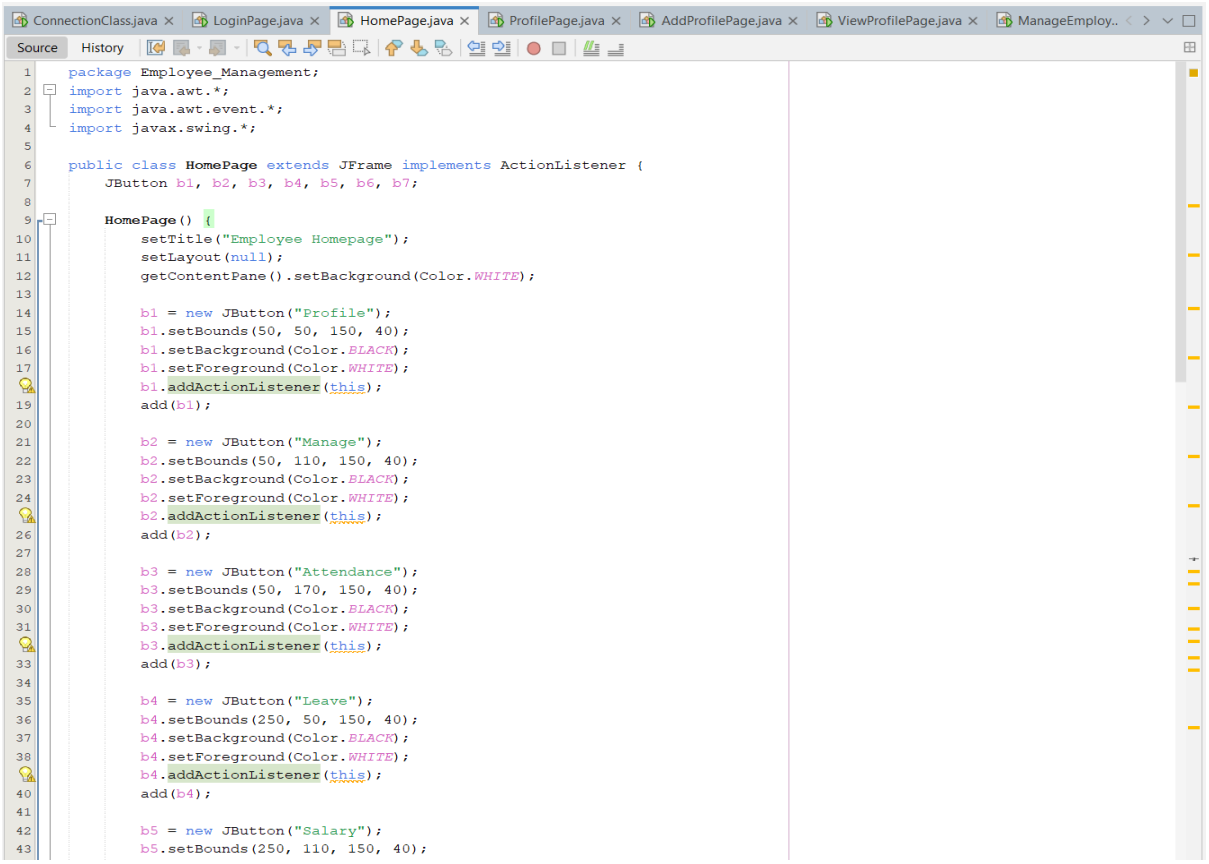




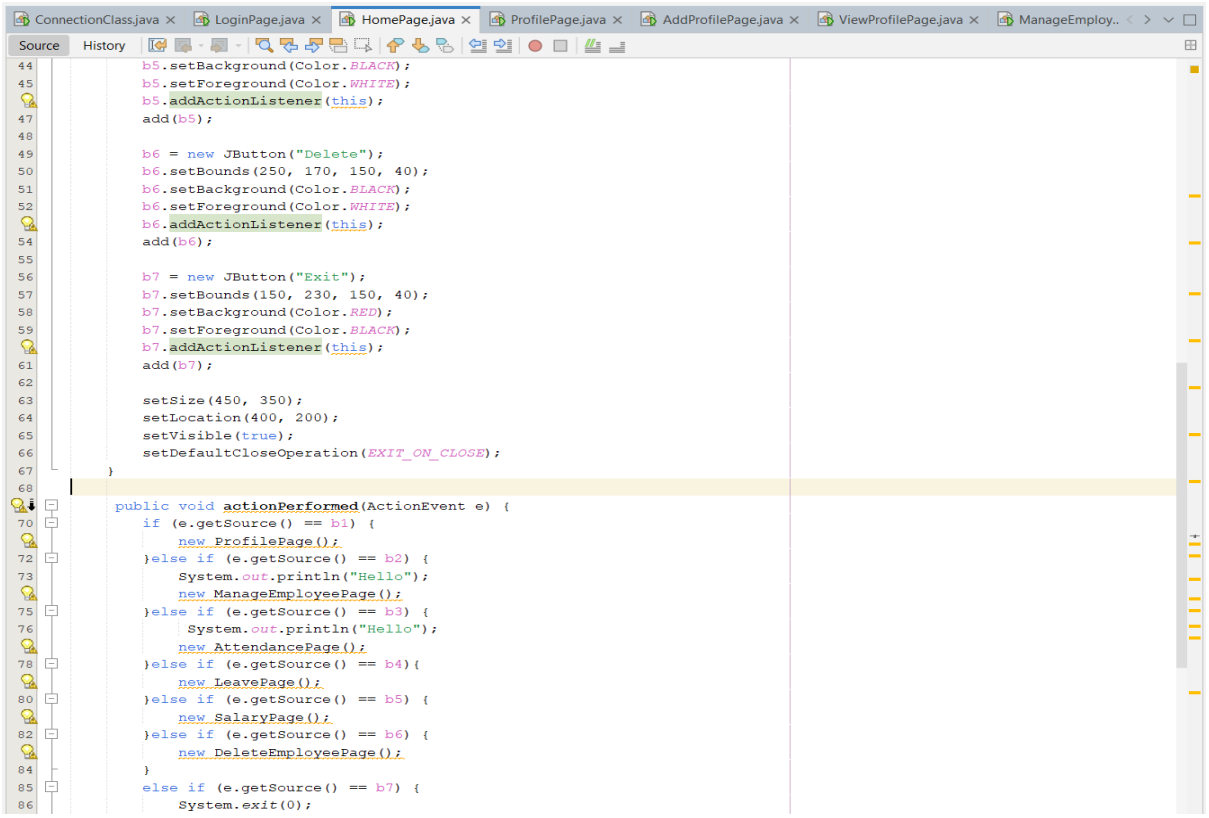

```java
package Employee_Management;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.*;
public class LoginPage extends JFrame implements ActionListener {
    JFrame f;
    JLabel l1,l2;
    JTextField t1;
    JPasswordField t2;
    JButton b1,b2;
    LoginPage(){
        f = new JFrame("Login:");
        f.setBackground(Color.WHITE);
        f.setLayout(null);

        l1 = new JLabel("Username");
        l1.setBounds(40, 20, 1000, 30);
        f.add(l1);

        l2 = new JLabel("Password");
        l2.setBounds(40, 70, 1000, 30);
        f.add(l2);

        t1 = new JTextField();;
        t1.setBounds(150, 20, 150, 30);
        f.add(t1);

        t2 = new JPasswordField();
        t2.setBounds(150, 70, 150, 30);
        f.add(t2);

        b1 = new JButton("Login");
        b1.setBackground(Color.BLACK);
        b1.setBounds(40, 140, 120, 30);
        b1.addActionListener(this);
        b1.setForeground(Color.WHITE);
        f.add(b1);

        b2 = new JButton("Close");
        b2.setBackground(Color.BLACK);
        b2.setBounds(180, 140, 120, 30);
        b2.addActionListener(this);
```



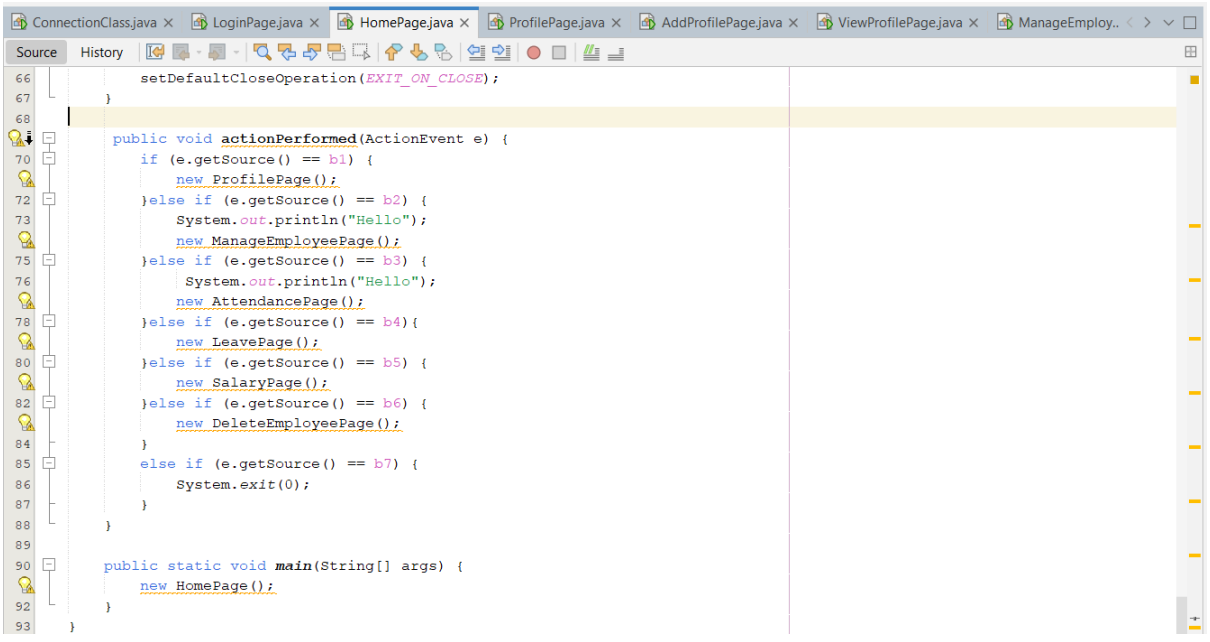
```
        b2.setForeground(Color.WHITE);
        f.add(b2);

        f.getContentPane();
        f.setVisible(true);
        f.setSize(500,240);
        f.setLocation(400,300);
    }
    public void actionPerformed(ActionEvent ee){
            if(ee.getSource() == b1){
                try{
                    ConnectionClass obj = new ConnectionClass();
                    String name = t1.getText();
                    String pass = t2.getText();
                    String q = "select * from logindata where username = '"+name+"' and password = '" +pass+"'";
                    ResultSet rs = obj.stm.executeQuery(q);
                    if(rs.next()){
                        new HomePage().setVisible(true);
                        this.setVisible(false);
                    }
                    else{
                        JOptionPane.showMessageDialog(null,"You entered wrong username and password");
                        f.setVisible(false);
                        f.setVisible(true);
                    }
                }
                catch(Exception e){
                    e.printStackTrace();
                }
            }
            if(ee.getSource() == b2){
                this.f.setVisible(false);
            }
    }
    public static void main(String args[]){
        new LoginPage();
    }
}
```



```
package Employee_Management;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class HomePage extends JFrame implements ActionListener {
    JButton b1, b2, b3, b4, b5, b6, b7;

    HomePage() {
        setTitle("Employee Homepage");
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        b1 = new JButton("Profile");
        b1.setBounds(50, 50, 150, 40);
        b1.setBackground(Color.BLACK);
        b1.setForeground(Color.WHITE);
        b1.addActionListener(this);
        add(b1);

        b2 = new JButton("Manage");
        b2.setBounds(50, 110, 150, 40);
        b2.setBackground(Color.BLACK);
        b2.setForeground(Color.WHITE);
        b2.addActionListener(this);
        add(b2);

        b3 = new JButton("Attendance");
        b3.setBounds(50, 170, 150, 40);
        b3.setBackground(Color.BLACK);
        b3.setForeground(Color.WHITE);
        b3.addActionListener(this);
        add(b3);

        b4 = new JButton("Leave");
        b4.setBounds(250, 50, 150, 40);
        b4.setBackground(Color.BLACK);
        b4.setForeground(Color.WHITE);
        b4.addActionListener(this);
        add(b4);

        b5 = new JButton("Salary");
        b5.setBounds(250, 110, 150, 40);
```



```
            b5.setBackground(Color.BLACK);
            b5.setForeground(Color.WHITE);
            b5.addActionListener(this);
            add(b5);

            b6 = new JButton("Delete");
            b6.setBounds(250, 170, 150, 40);
            b6.setBackground(Color.BLACK);
            b6.setForeground(Color.WHITE);
            b6.addActionListener(this);
            add(b6);

            b7 = new JButton("Exit");
            b7.setBounds(150, 230, 150, 40);
            b7.setBackground(Color.RED);
            b7.setForeground(Color.BLACK);
            b7.addActionListener(this);
            add(b7);

            setSize(450, 350);
            setLocation(400, 200);
            setVisible(true);
            setDefaultCloseOperation(EXIT_ON_CLOSE);
        }

        public void actionPerformed(ActionEvent e) {
            if (e.getSource() == b1) {
                new ProfilePage();
            }else if (e.getSource() == b2) {
                System.out.println("Hello");
                new ManageEmployeePage();
            }else if (e.getSource() == b3) {
                 System.out.println("Hello");
                new AttendancePage();
            }else if (e.getSource() == b4){
                new LeavePage();
            }else if (e.getSource() == b5) {
                new SalaryPage();
            }else if (e.getSource() == b6) {
                new DeleteEmployeePage();
            }
            else if (e.getSource() == b7) {
                System.exit(0);
```

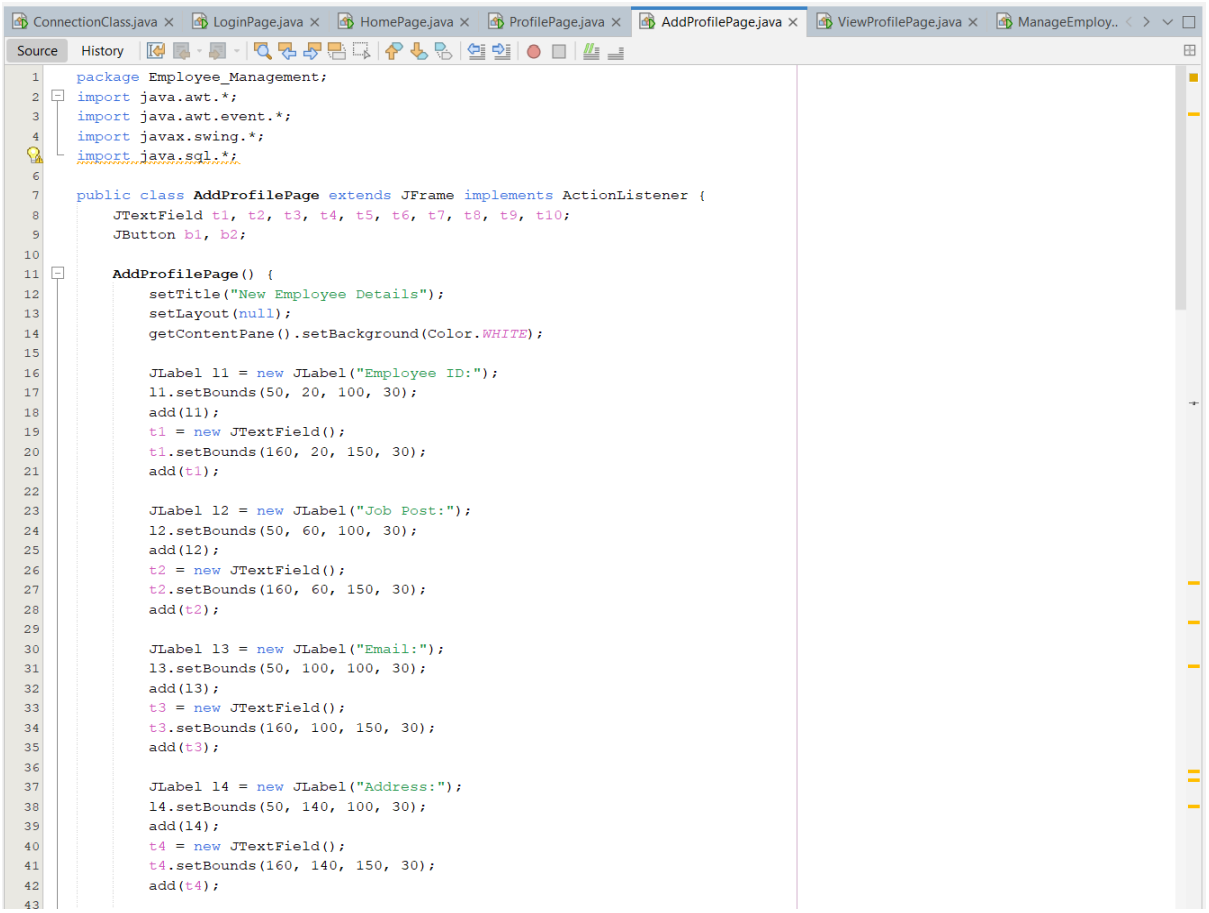


```
            setDefaultCloseOperation(EXIT_ON_CLOSE);
        }

        public void actionPerformed(ActionEvent e) {
            if (e.getSource() == b1) {
                new ProfilePage();
            }else if (e.getSource() == b2) {
                System.out.println("Hello");
                new ManageEmployeePage();
            }else if (e.getSource() == b3) {
                 System.out.println("Hello");
                new AttendancePage();
            }else if (e.getSource() == b4){
                new LeavePage();
            }else if (e.getSource() == b5) {
                new SalaryPage();
            }else if (e.getSource() == b6) {
                new DeleteEmployeePage();
            }
            else if (e.getSource() == b7) {
                System.exit(0);
            }
        }

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

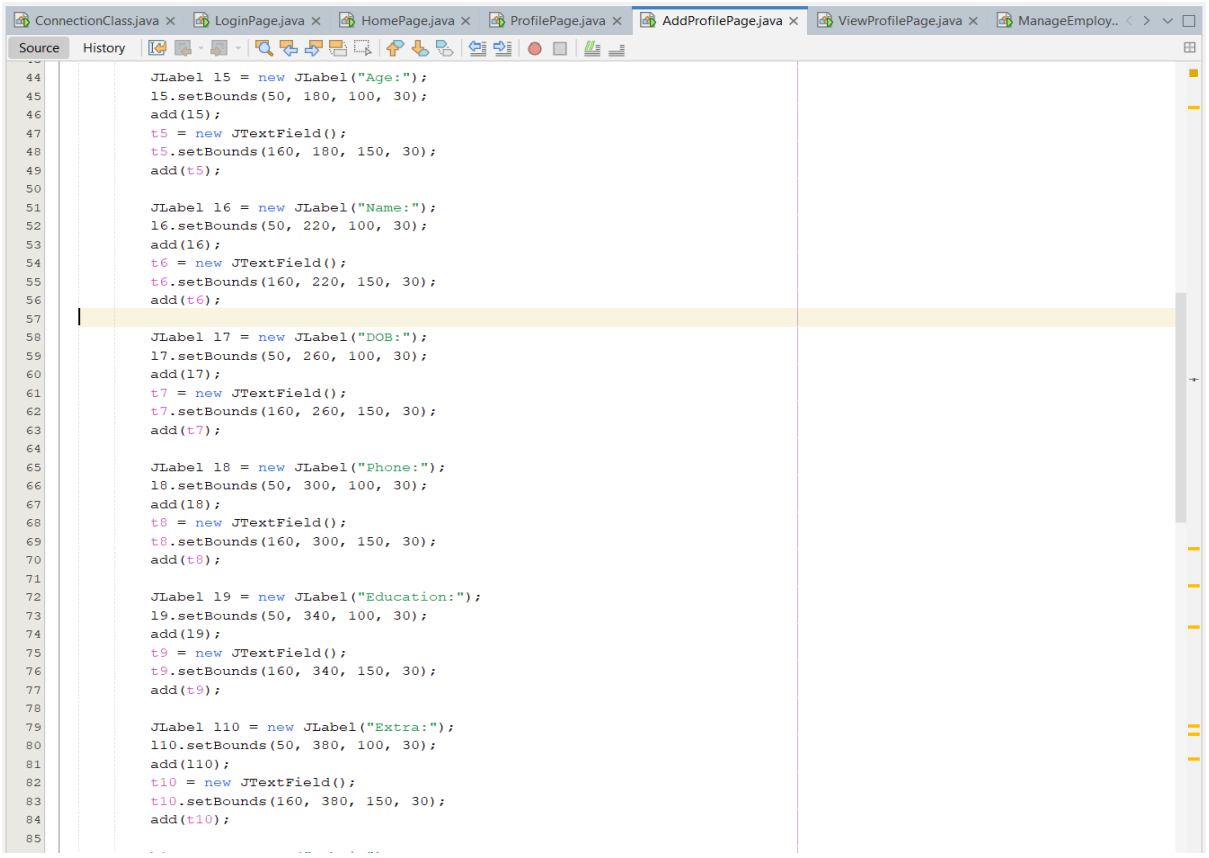
ConnectionClass.java | LoginPage.java | HomePage.java | ProfilePage.java | AddProfilePage.java | ViewProfilePage.java | ManageEmploy..

Source | History

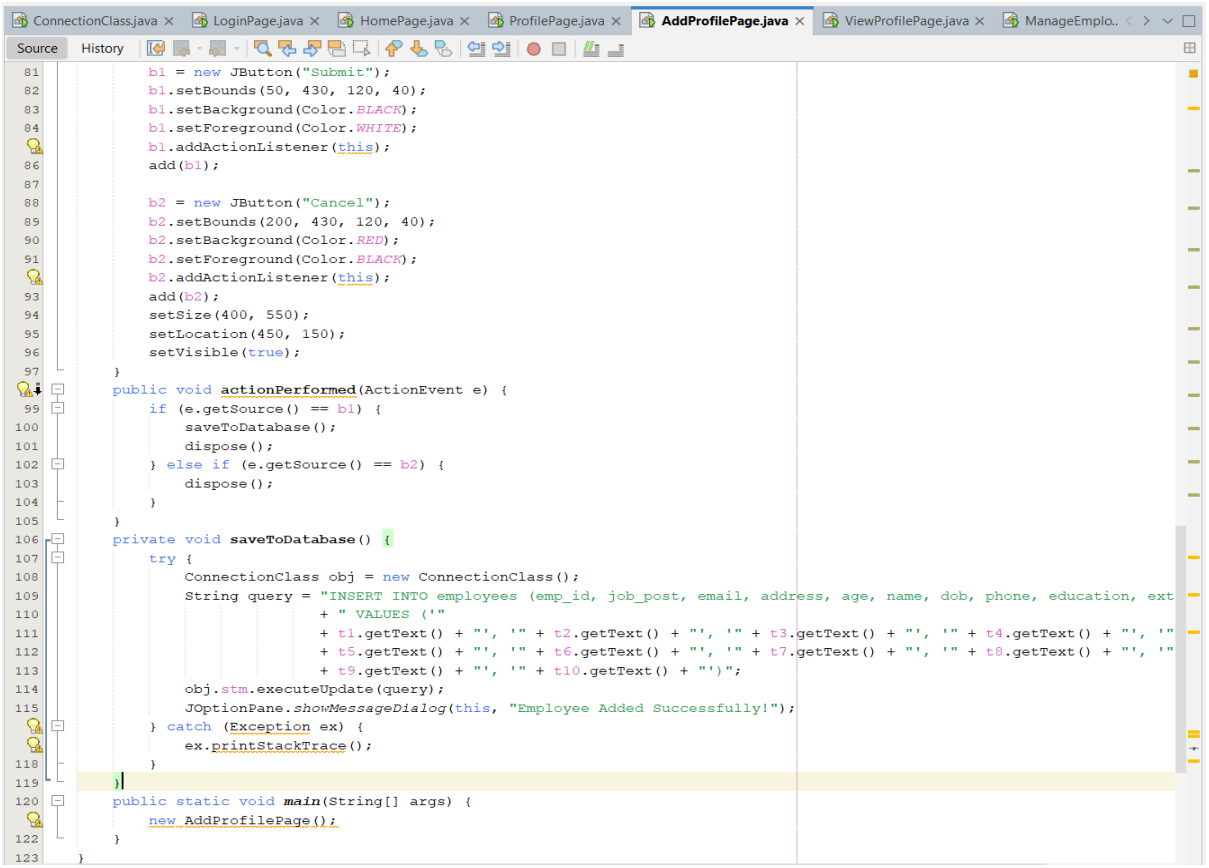
```java
package Employee_Management;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ProfilePage extends JFrame implements ActionListener {
    JButton b1, b2;
    ProfilePage() {
        setTitle("Profile Options");
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        b1 = new JButton("View Profile");
        b1.setBounds(50, 50, 150, 40);
        b1.setBackground(Color.BLACK);
        b1.setForeground(Color.WHITE);
        b1.addActionListener(this);
        add(b1);

        b2 = new JButton("Add Profile");
        b2.setBounds(250, 50, 150, 40);
        b2.setBackground(Color.BLACK);
        b2.setForeground(Color.WHITE);
        b2.addActionListener(this);
        add(b2);

        setSize(450, 200);
        setLocation(400, 250);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == b1) {
            new ViewProfilePage();
        } else if (e.getSource() == b2) {
            new AddProfilePage();
        }
    }

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



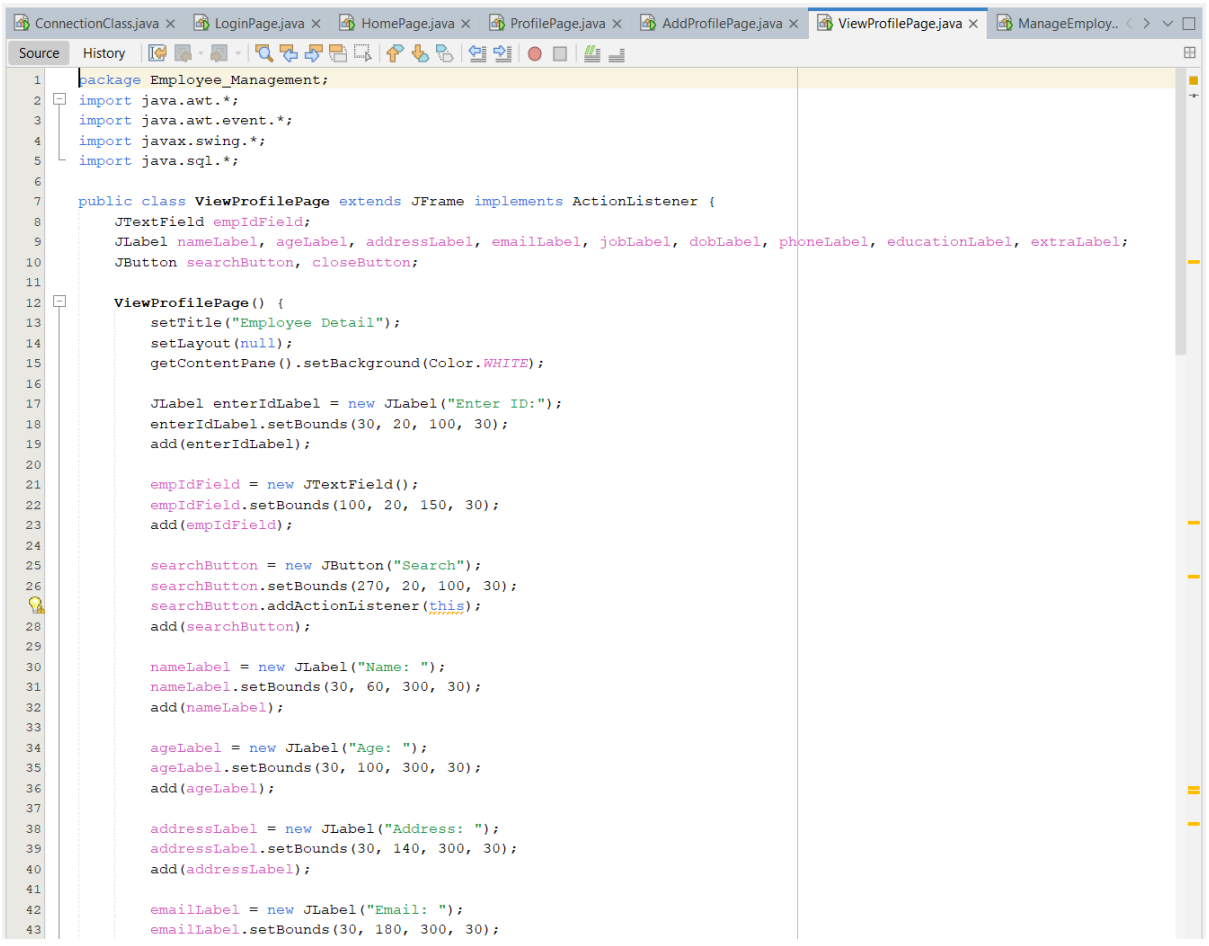

```java
package Employee_Management;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class AddProfilePage extends JFrame implements ActionListener {
    JTextField t1, t2, t3, t4, t5, t6, t7, t8, t9, t10;
    JButton b1, b2;

    AddProfilePage() {
        setTitle("New Employee Details");
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        JLabel l1 = new JLabel("Employee ID:");
        l1.setBounds(50, 20, 100, 30);
        add(l1);
        t1 = new JTextField();
        t1.setBounds(160, 20, 150, 30);
        add(t1);

        JLabel l2 = new JLabel("Job Post:");
        l2.setBounds(50, 60, 100, 30);
        add(l2);
        t2 = new JTextField();
        t2.setBounds(160, 60, 150, 30);
        add(t2);

        JLabel l3 = new JLabel("Email:");
        l3.setBounds(50, 100, 100, 30);
        add(l3);
        t3 = new JTextField();
        t3.setBounds(160, 100, 150, 30);
        add(t3);

        JLabel l4 = new JLabel("Address:");
        l4.setBounds(50, 140, 100, 30);
        add(l4);
        t4 = new JTextField();
        t4.setBounds(160, 140, 150, 30);
        add(t4);
```



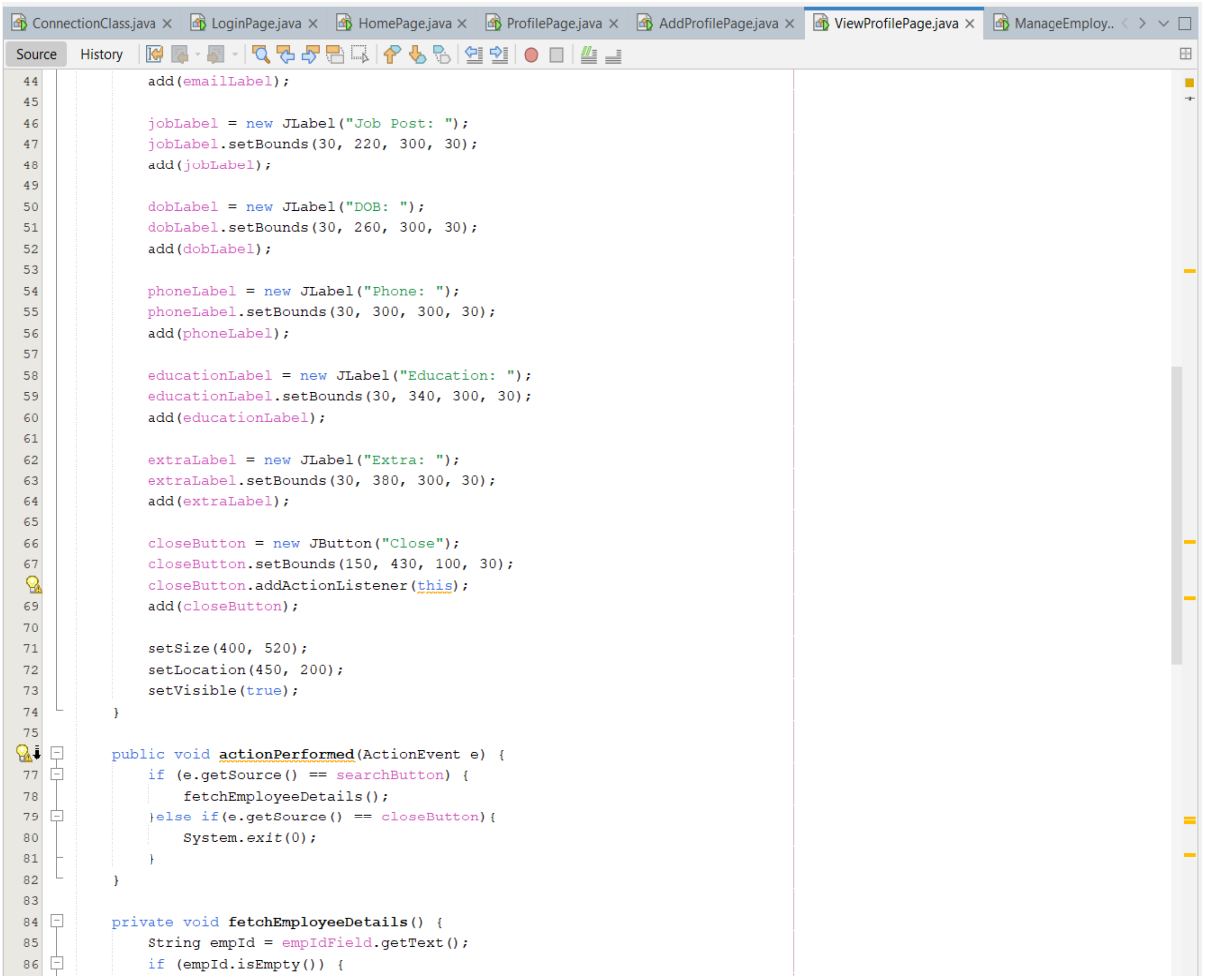

```java
        JLabel l5 = new JLabel("Age:");
        l5.setBounds(50, 180, 100, 30);
        add(l5);
        t5 = new JTextField();
        t5.setBounds(160, 180, 150, 30);
        add(t5);

        JLabel l6 = new JLabel("Name:");
        l6.setBounds(50, 220, 100, 30);
        add(l6);
        t6 = new JTextField();
        t6.setBounds(160, 220, 150, 30);
        add(t6);

        JLabel l7 = new JLabel("DOB:");
        l7.setBounds(50, 260, 100, 30);
        add(l7);
        t7 = new JTextField();
        t7.setBounds(160, 260, 150, 30);
        add(t7);

        JLabel l8 = new JLabel("Phone:");
        l8.setBounds(50, 300, 100, 30);
        add(l8);
        t8 = new JTextField();
        t8.setBounds(160, 300, 150, 30);
        add(t8);

        JLabel l9 = new JLabel("Education:");
        l9.setBounds(50, 340, 100, 30);
        add(l9);
        t9 = new JTextField();
        t9.setBounds(160, 340, 150, 30);
        add(t9);

        JLabel l10 = new JLabel("Extra:");
        l10.setBounds(50, 380, 100, 30);
        add(l10);
        t10 = new JTextField();
        t10.setBounds(160, 380, 150, 30);
        add(t10);
```



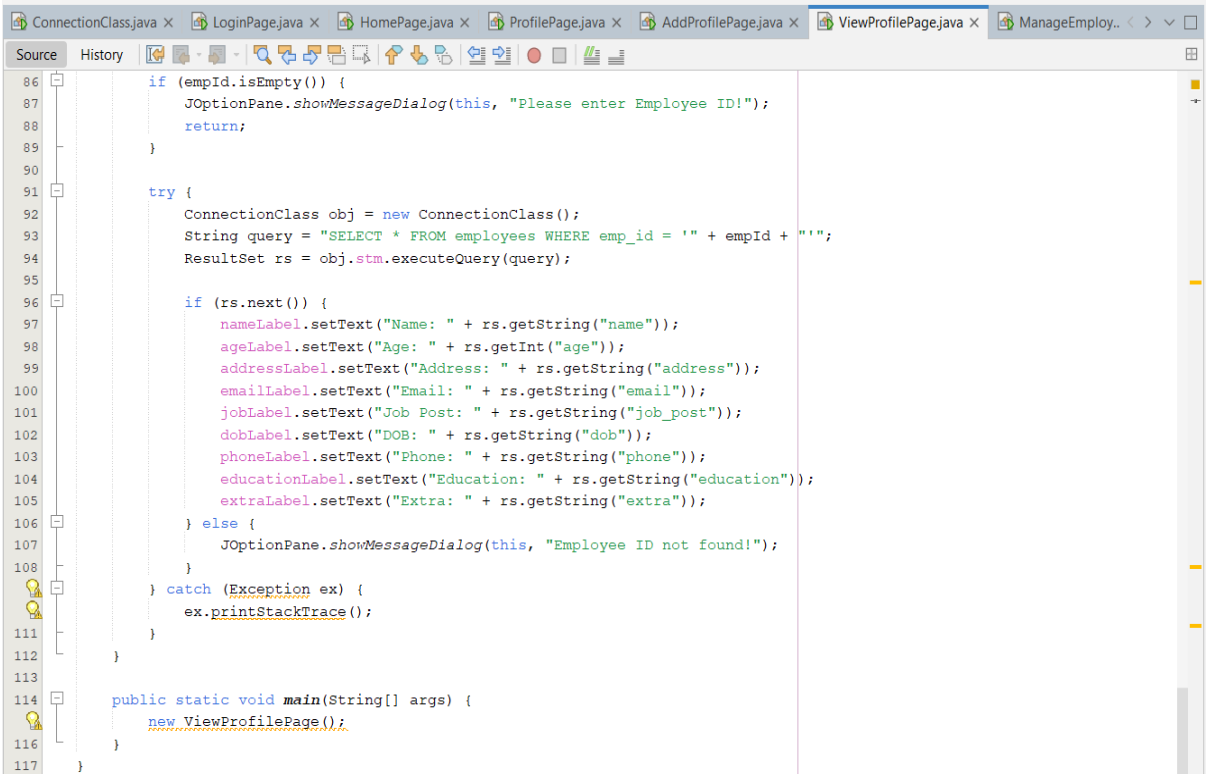
```
        b1 = new JButton("Submit");
        b1.setBounds(50, 430, 120, 40);
        b1.setBackground(Color.BLACK);
        b1.setForeground(Color.WHITE);
        b1.addActionListener(this);
        add(b1);

        b2 = new JButton("Cancel");
        b2.setBounds(200, 430, 120, 40);
        b2.setBackground(Color.RED);
        b2.setForeground(Color.BLACK);
        b2.addActionListener(this);
        add(b2);
        setSize(400, 550);
        setLocation(450, 150);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == b1) {
            saveToDatabase();
            dispose();
        } else if (e.getSource() == b2) {
            dispose();
        }
    }
    private void saveToDatabase() {
        try {
            ConnectionClass obj = new ConnectionClass();
            String query = "INSERT INTO employees (emp_id, job_post, email, address, age, name, dob, phone, education, ext
                         + " VALUES ('"
                         + t1.getText() + "', '" + t2.getText() + "', '" + t3.getText() + "', '" + t4.getText() + "', '"
                         + t5.getText() + "', '" + t6.getText() + "', '" + t7.getText() + "', '" + t8.getText() + "', '"
                         + t9.getText() + "', '" + t10.getText() + "')";
            obj.stm.executeUpdate(query);
            JOptionPane.showMessageDialog(this, "Employee Added Successfully!");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    public static void main(String[] args) {
        new AddProfilePage();
    }
}
```



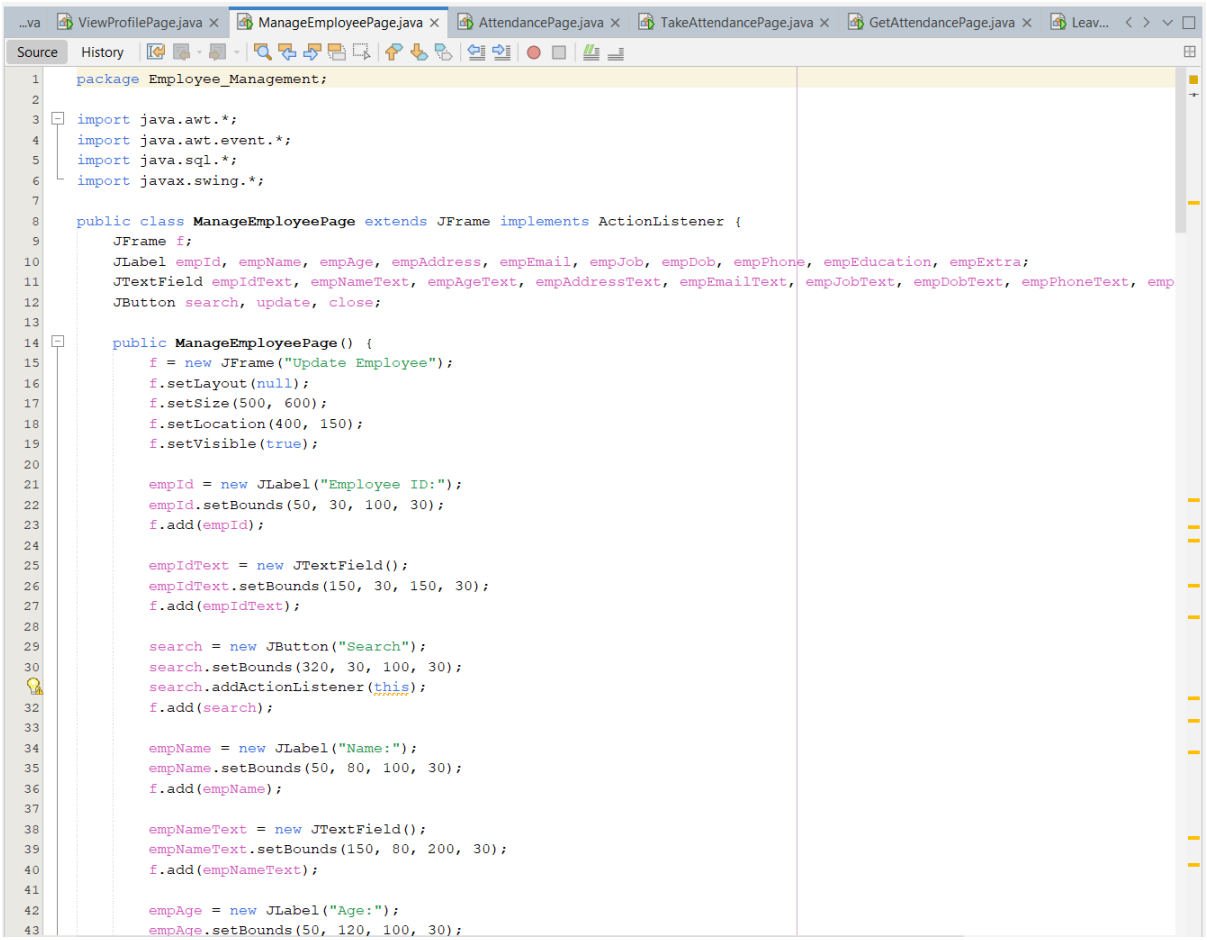
```
package Employee_Management;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class ViewProfilePage extends JFrame implements ActionListener {
    JTextField empIdField;
    JLabel nameLabel, ageLabel, addressLabel, emailLabel, jobLabel, dobLabel, phoneLabel, educationLabel, extraLabel;
    JButton searchButton, closeButton;

    ViewProfilePage() {
        setTitle("Employee Detail");
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        JLabel enterIdLabel = new JLabel("Enter ID:");
        enterIdLabel.setBounds(30, 20, 100, 30);
        add(enterIdLabel);

        empIdField = new JTextField();
        empIdField.setBounds(100, 20, 150, 30);
        add(empIdField);

        searchButton = new JButton("Search");
        searchButton.setBounds(270, 20, 100, 30);
        searchButton.addActionListener(this);
        add(searchButton);

        nameLabel = new JLabel("Name: ");
        nameLabel.setBounds(30, 60, 300, 30);
        add(nameLabel);

        ageLabel = new JLabel("Age: ");
        ageLabel.setBounds(30, 100, 300, 30);
        add(ageLabel);

        addressLabel = new JLabel("Address: ");
        addressLabel.setBounds(30, 140, 300, 30);
        add(addressLabel);

        emailLabel = new JLabel("Email: ");
        emailLabel.setBounds(30, 180, 300, 30);
```



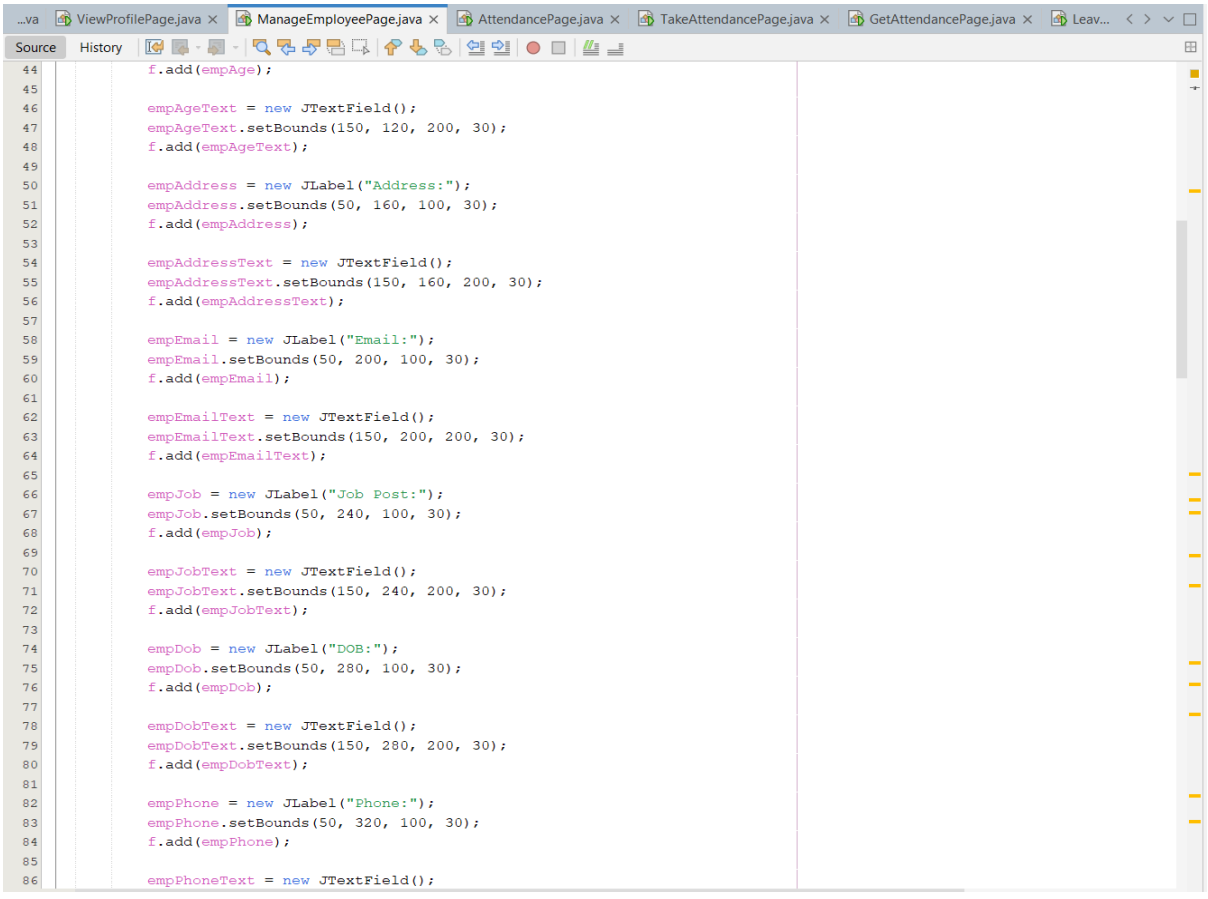
```
        add(emailLabel);

        jobLabel = new JLabel("Job Post: ");
        jobLabel.setBounds(30, 220, 300, 30);
        add(jobLabel);

        dobLabel = new JLabel("DOB: ");
        dobLabel.setBounds(30, 260, 300, 30);
        add(dobLabel);

        phoneLabel = new JLabel("Phone: ");
        phoneLabel.setBounds(30, 300, 300, 30);
        add(phoneLabel);

        educationLabel = new JLabel("Education: ");
        educationLabel.setBounds(30, 340, 300, 30);
        add(educationLabel);

        extraLabel = new JLabel("Extra: ");
        extraLabel.setBounds(30, 380, 300, 30);
        add(extraLabel);

        closeButton = new JButton("Close");
        closeButton.setBounds(150, 430, 100, 30);
        closeButton.addActionListener(this);
        add(closeButton);

        setSize(400, 520);
        setLocation(450, 200);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == searchButton) {
            fetchEmployeeDetails();
        }else if(e.getSource() == closeButton){
            System.exit(0);
        }
    }

    private void fetchEmployeeDetails() {
        String empId = empIdField.getText();
        if (empId.isEmpty()) {
```



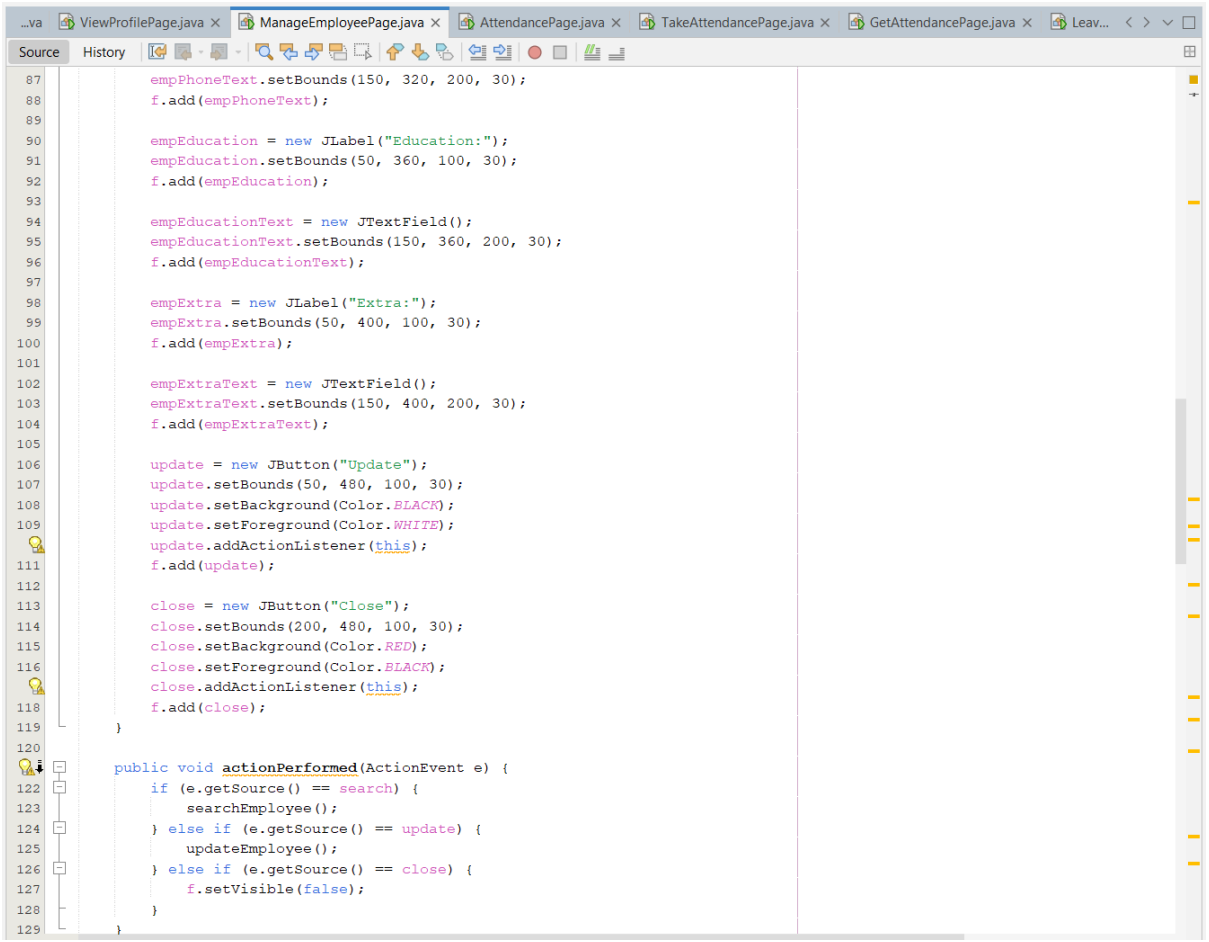
```
        if (empId.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Please enter Employee ID!");
            return;
        }

        try {
            ConnectionClass obj = new ConnectionClass();
            String query = "SELECT * FROM employees WHERE emp_id = '" + empId + "'";
            ResultSet rs = obj.stm.executeQuery(query);

            if (rs.next()) {
                nameLabel.setText("Name: " + rs.getString("name"));
                ageLabel.setText("Age: " + rs.getInt("age"));
                addressLabel.setText("Address: " + rs.getString("address"));
                emailLabel.setText("Email: " + rs.getString("email"));
                jobLabel.setText("Job Post: " + rs.getString("job_post"));
                dobLabel.setText("DOB: " + rs.getString("dob"));
                phoneLabel.setText("Phone: " + rs.getString("phone"));
                educationLabel.setText("Education: " + rs.getString("education"));
                extraLabel.setText("Extra: " + rs.getString("extra"));
            } else {
                JOptionPane.showMessageDialog(this, "Employee ID not found!");
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

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



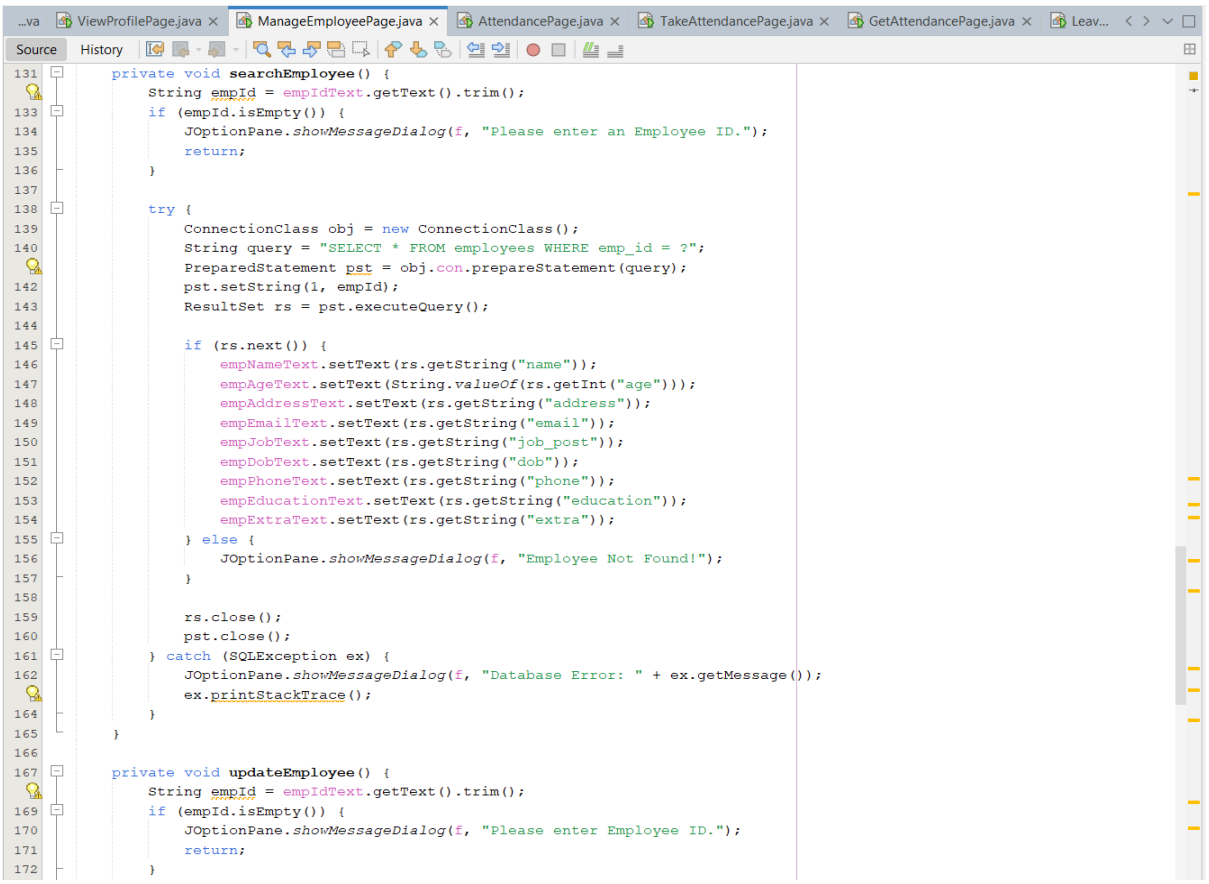

```java
package Employee_Management;

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

public class ManageEmployeePage extends JFrame implements ActionListener {
    JFrame f;
    JLabel empId, empName, empAge, empAddress, empEmail, empJob, empDob, empPhone, empEducation, empExtra;
    JTextField empIdText, empNameText, empAgeText, empAddressText, empEmailText, empJobText, empDobText, empPhoneText, emp
    JButton search, update, close;

    public ManageEmployeePage() {
        f = new JFrame("Update Employee");
        f.setLayout(null);
        f.setSize(500, 600);
        f.setLocation(400, 150);
        f.setVisible(true);

        empId = new JLabel("Employee ID:");
        empId.setBounds(50, 30, 100, 30);
        f.add(empId);

        empIdText = new JTextField();
        empIdText.setBounds(150, 30, 150, 30);
        f.add(empIdText);

        search = new JButton("Search");
        search.setBounds(320, 30, 100, 30);
        search.addActionListener(this);
        f.add(search);

        empName = new JLabel("Name:");
        empName.setBounds(50, 80, 100, 30);
        f.add(empName);

        empNameText = new JTextField();
        empNameText.setBounds(150, 80, 200, 30);
        f.add(empNameText);

        empAge = new JLabel("Age:");
        empAge.setBounds(50, 120, 100, 30);
```



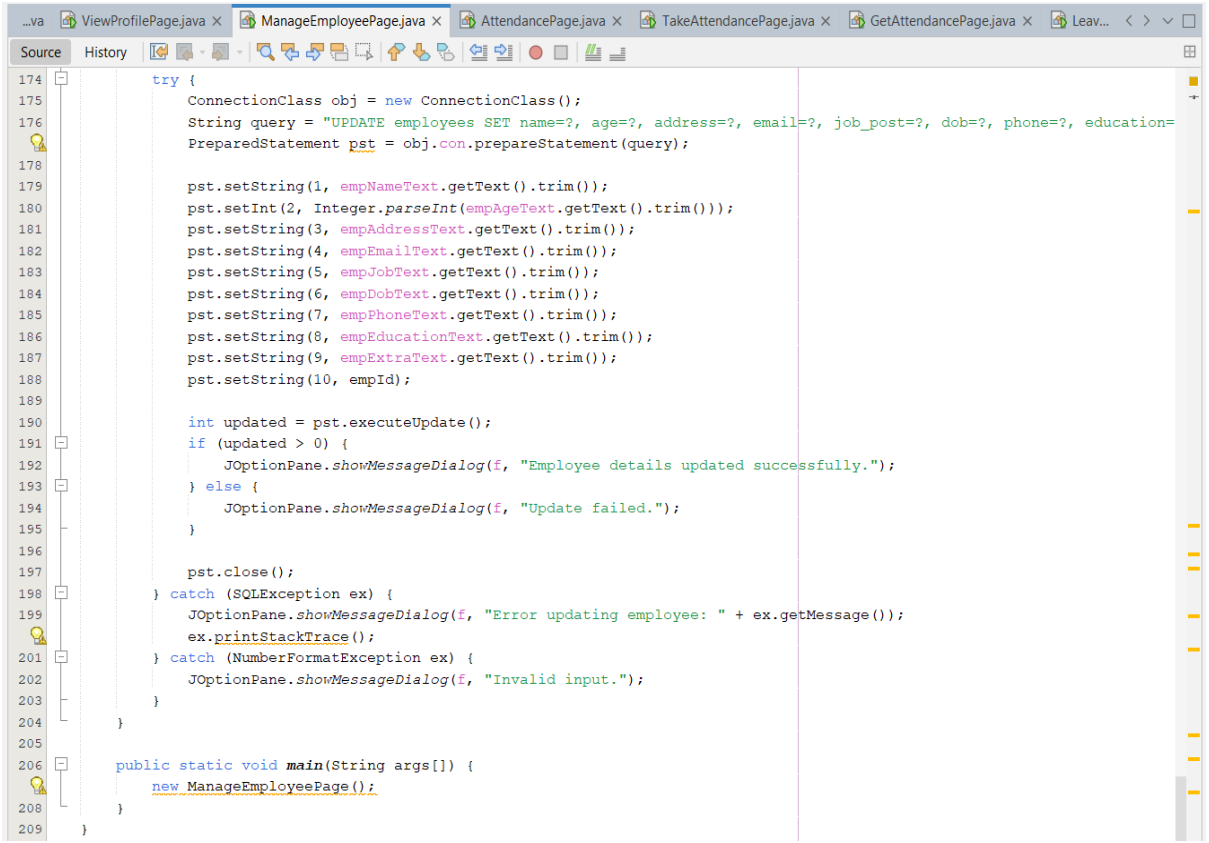

```java
        f.add(empAge);

        empAgeText = new JTextField();
        empAgeText.setBounds(150, 120, 200, 30);
        f.add(empAgeText);

        empAddress = new JLabel("Address:");
        empAddress.setBounds(50, 160, 100, 30);
        f.add(empAddress);

        empAddressText = new JTextField();
        empAddressText.setBounds(150, 160, 200, 30);
        f.add(empAddressText);

        empEmail = new JLabel("Email:");
        empEmail.setBounds(50, 200, 100, 30);
        f.add(empEmail);

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

        empJob = new JLabel("Job Post:");
        empJob.setBounds(50, 240, 100, 30);
        f.add(empJob);

        empJobText = new JTextField();
        empJobText.setBounds(150, 240, 200, 30);
        f.add(empJobText);

        empDob = new JLabel("DOB:");
        empDob.setBounds(50, 280, 100, 30);
        f.add(empDob);

        empDobText = new JTextField();
        empDobText.setBounds(150, 280, 200, 30);
        f.add(empDobText);

        empPhone = new JLabel("Phone:");
        empPhone.setBounds(50, 320, 100, 30);
        f.add(empPhone);

        empPhoneText = new JTextField();
```



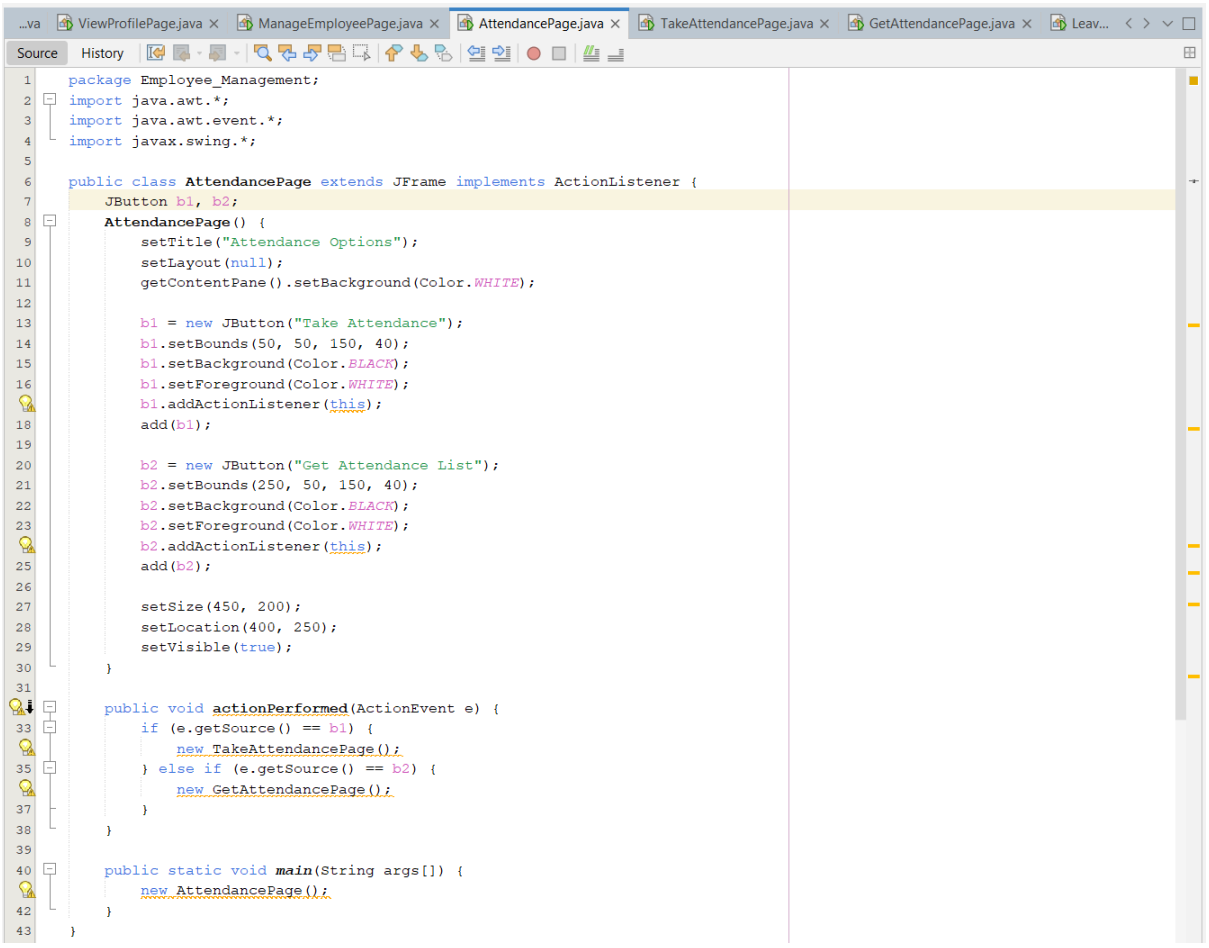
```
87          empPhoneText.setBounds(150, 320, 200, 30);
88          f.add(empPhoneText);
89
90          empEducation = new JLabel("Education:");
91          empEducation.setBounds(50, 360, 100, 30);
92          f.add(empEducation);
93
94          empEducationText = new JTextField();
95          empEducationText.setBounds(150, 360, 200, 30);
96          f.add(empEducationText);
97
98          empExtra = new JLabel("Extra:");
99          empExtra.setBounds(50, 400, 100, 30);
100         f.add(empExtra);
101
102         empExtraText = new JTextField();
103         empExtraText.setBounds(150, 400, 200, 30);
104         f.add(empExtraText);
105
106         update = new JButton("Update");
107         update.setBounds(50, 480, 100, 30);
108         update.setBackground(Color.BLACK);
109         update.setForeground(Color.WHITE);
110         update.addActionListener(this);
111         f.add(update);
112
113         close = new JButton("Close");
114         close.setBounds(200, 480, 100, 30);
115         close.setBackground(Color.RED);
116         close.setForeground(Color.BLACK);
117         close.addActionListener(this);
118         f.add(close);
119     }
120
121     public void actionPerformed(ActionEvent e) {
122         if (e.getSource() == search) {
123             searchEmployee();
124         } else if (e.getSource() == update) {
125             updateEmployee();
126         } else if (e.getSource() == close) {
127             f.setVisible(false);
128         }
129     }
```



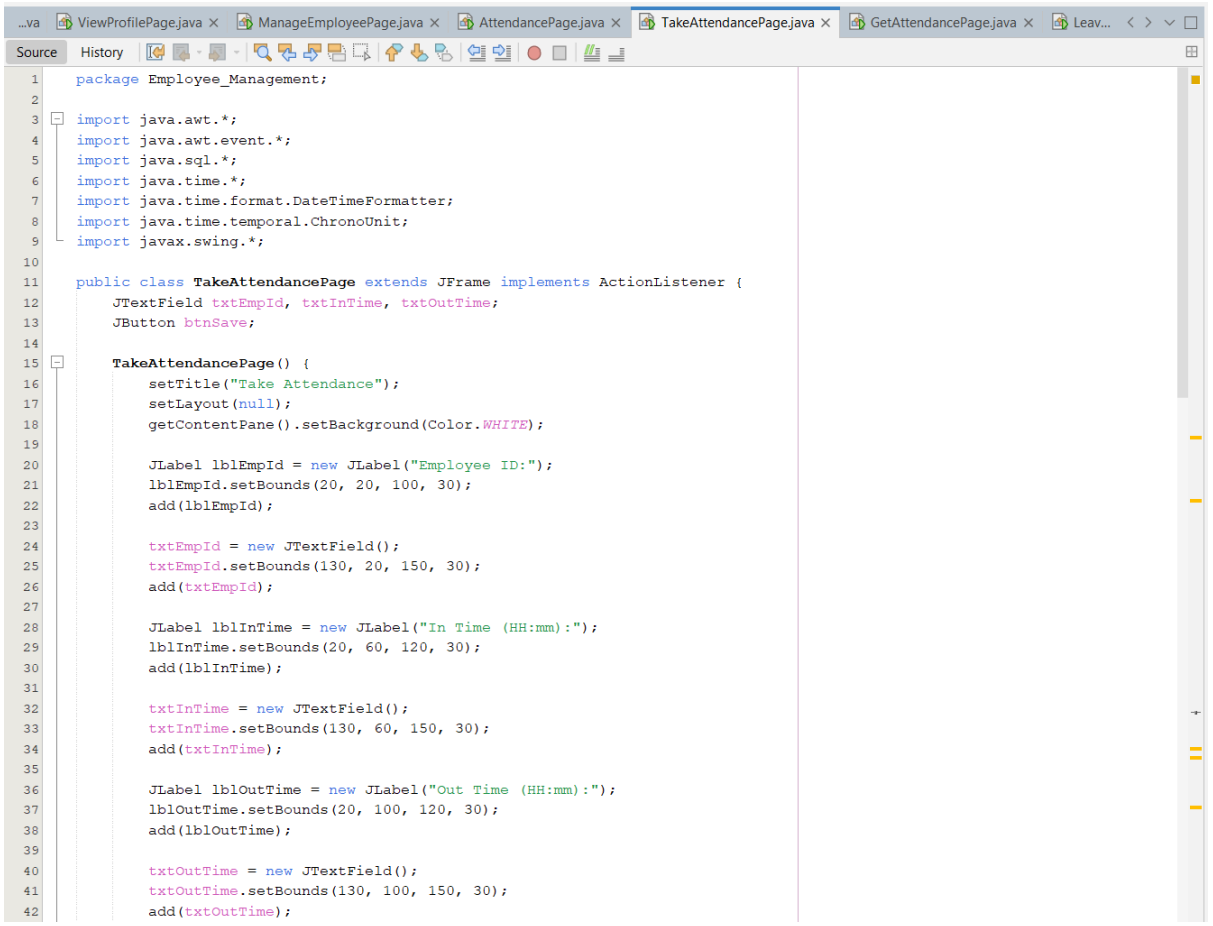
```
131     private void searchEmployee() {
132         String empId = empIdText.getText().trim();
133         if (empId.isEmpty()) {
134             JOptionPane.showMessageDialog(f, "Please enter an Employee ID.");
135             return;
136         }
137
138         try {
139             ConnectionClass obj = new ConnectionClass();
140             String query = "SELECT * FROM employees WHERE emp_id = ?";
141             PreparedStatement pst = obj.con.prepareStatement(query);
142             pst.setString(1, empId);
143             ResultSet rs = pst.executeQuery();
144
145             if (rs.next()) {
146                 empNameText.setText(rs.getString("name"));
147                 empAgeText.setText(String.valueOf(rs.getInt("age")));
148                 empAddressText.setText(rs.getString("address"));
149                 empEmailText.setText(rs.getString("email"));
150                 empJobText.setText(rs.getString("job_post"));
151                 empDobText.setText(rs.getString("dob"));
152                 empPhoneText.setText(rs.getString("phone"));
153                 empEducationText.setText(rs.getString("education"));
154                 empExtraText.setText(rs.getString("extra"));
155             } else {
156                 JOptionPane.showMessageDialog(f, "Employee Not Found!");
157             }
158
159             rs.close();
160             pst.close();
161         } catch (SQLException ex) {
162             JOptionPane.showMessageDialog(f, "Database Error: " + ex.getMessage());
163             ex.printStackTrace();
164         }
165     }
166
167     private void updateEmployee() {
168         String empId = empIdText.getText().trim();
169         if (empId.isEmpty()) {
170             JOptionPane.showMessageDialog(f, "Please enter Employee ID.");
171             return;
172         }
```



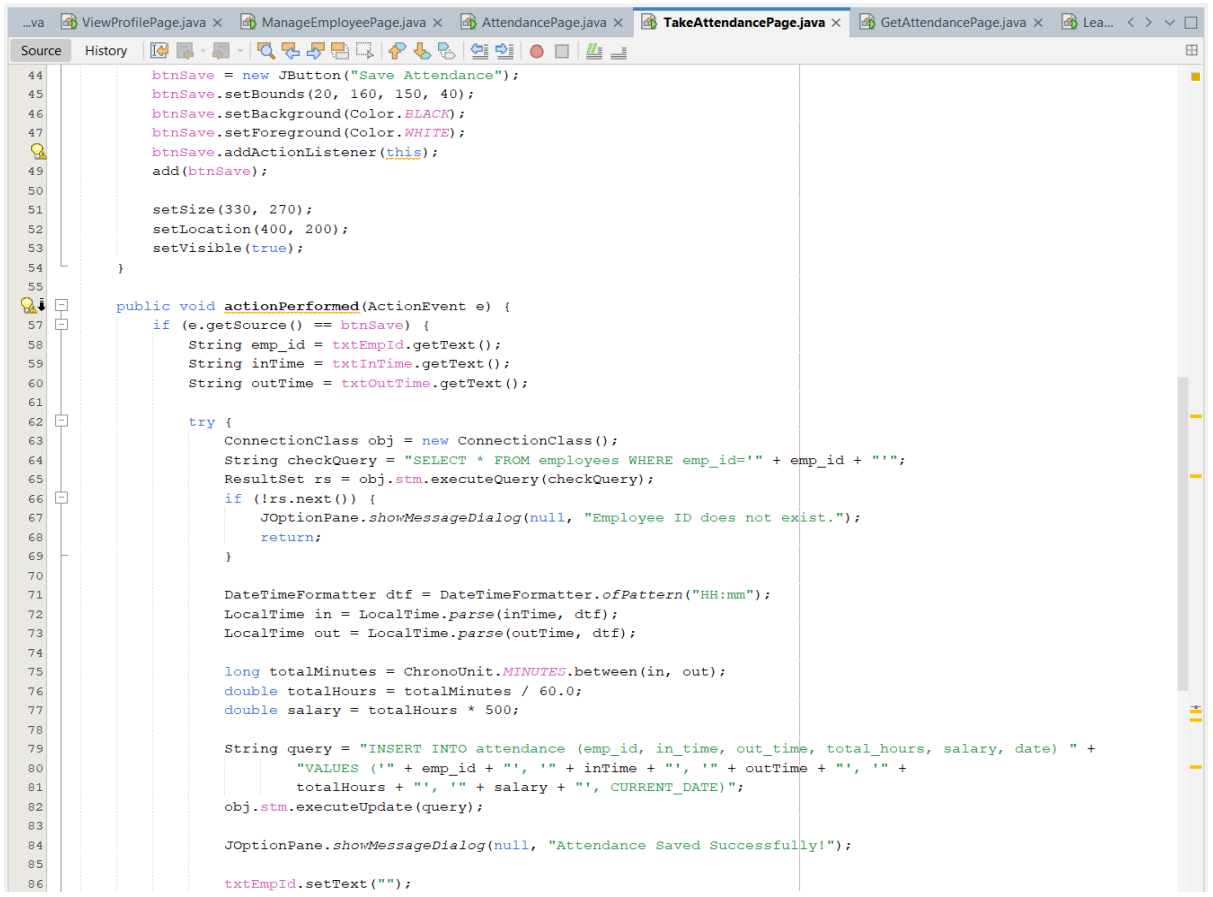
```
        try {
            ConnectionClass obj = new ConnectionClass();
            String query = "UPDATE employees SET name=?, age=?, address=?, email=?, job_post=?, dob=?, phone=?, education=
            PreparedStatement pst = obj.con.prepareStatement(query);

            pst.setString(1, empNameText.getText().trim());
            pst.setInt(2, Integer.parseInt(empAgeText.getText().trim()));
            pst.setString(3, empAddressText.getText().trim());
            pst.setString(4, empEmailText.getText().trim());
            pst.setString(5, empJobText.getText().trim());
            pst.setString(6, empDobText.getText().trim());
            pst.setString(7, empPhoneText.getText().trim());
            pst.setString(8, empEducationText.getText().trim());
            pst.setString(9, empExtraText.getText().trim());
            pst.setString(10, empId);

            int updated = pst.executeUpdate();
            if (updated > 0) {
                JOptionPane.showMessageDialog(f, "Employee details updated successfully.");
            } else {
                JOptionPane.showMessageDialog(f, "Update failed.");
            }

            pst.close();
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(f, "Error updating employee: " + ex.getMessage());
            ex.printStackTrace();
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(f, "Invalid input.");
        }
    }

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



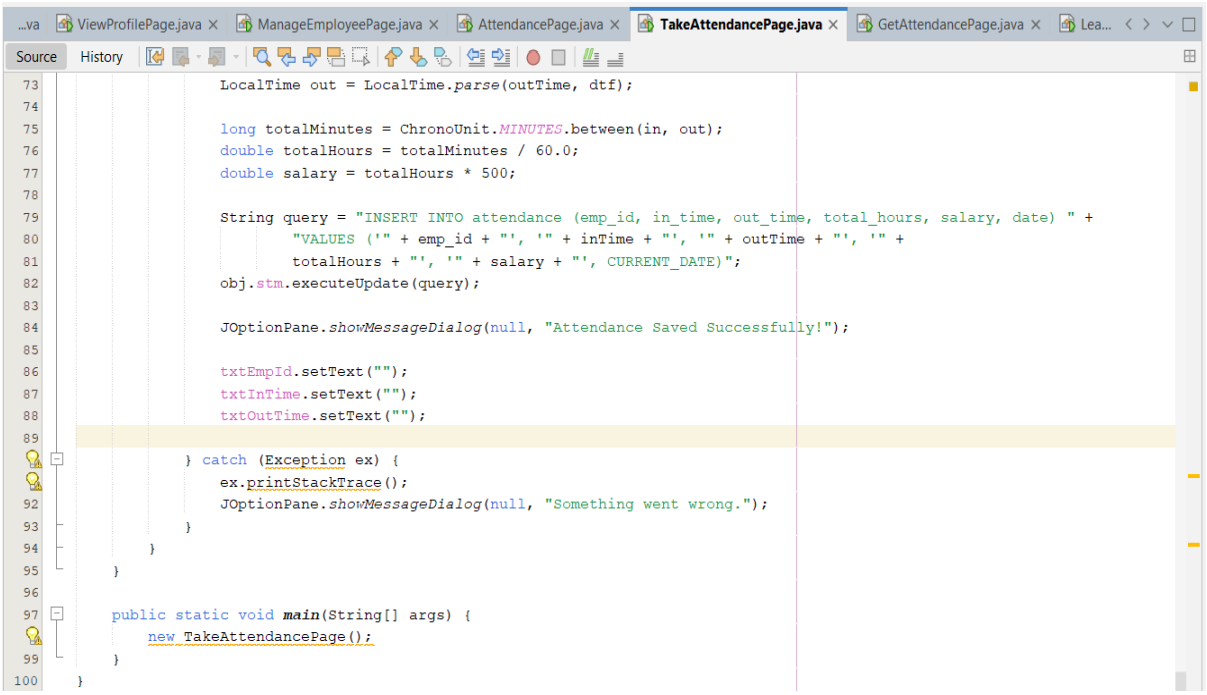
```
package Employee_Management;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class AttendancePage extends JFrame implements ActionListener {
    JButton b1, b2;
    AttendancePage() {
        setTitle("Attendance Options");
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        b1 = new JButton("Take Attendance");
        b1.setBounds(50, 50, 150, 40);
        b1.setBackground(Color.BLACK);
        b1.setForeground(Color.WHITE);
        b1.addActionListener(this);
        add(b1);

        b2 = new JButton("Get Attendance List");
        b2.setBounds(250, 50, 150, 40);
        b2.setBackground(Color.BLACK);
        b2.setForeground(Color.WHITE);
        b2.addActionListener(this);
        add(b2);

        setSize(450, 200);
        setLocation(400, 250);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == b1) {
            new TakeAttendancePage();
        } else if (e.getSource() == b2) {
            new GetAttendancePage();
        }
    }

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



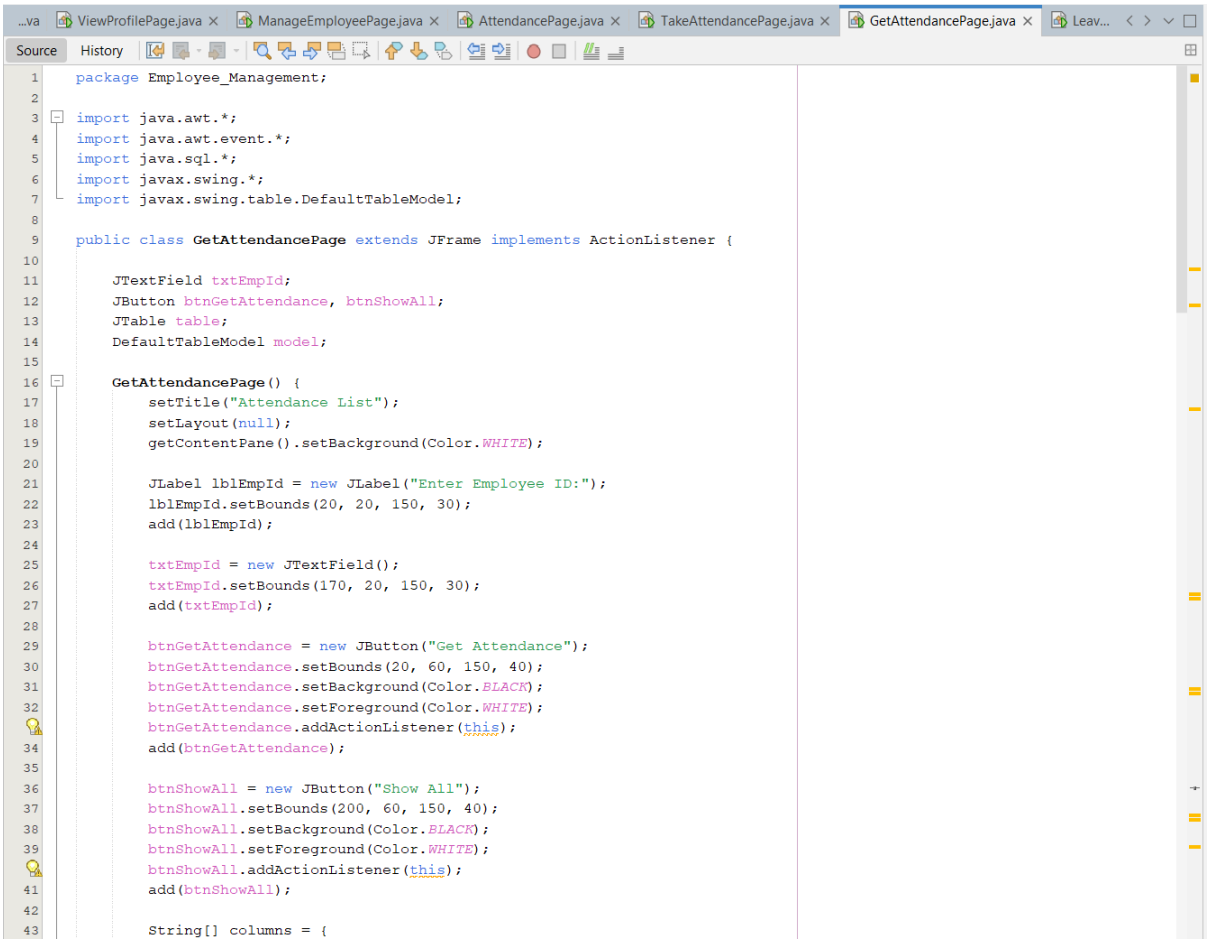
```java
package Employee_Management;

import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.time.*;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import javax.swing.*;

public class TakeAttendancePage extends JFrame implements ActionListener {
    JTextField txtEmpId, txtInTime, txtOutTime;
    JButton btnSave;

    TakeAttendancePage() {
        setTitle("Take Attendance");
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        JLabel lblEmpId = new JLabel("Employee ID:");
        lblEmpId.setBounds(20, 20, 100, 30);
        add(lblEmpId);

        txtEmpId = new JTextField();
        txtEmpId.setBounds(130, 20, 150, 30);
        add(txtEmpId);

        JLabel lblInTime = new JLabel("In Time (HH:mm):");
        lblInTime.setBounds(20, 60, 120, 30);
        add(lblInTime);

        txtInTime = new JTextField();
        txtInTime.setBounds(130, 60, 150, 30);
        add(txtInTime);

        JLabel lblOutTime = new JLabel("Out Time (HH:mm):");
        lblOutTime.setBounds(20, 100, 120, 30);
        add(lblOutTime);

        txtOutTime = new JTextField();
        txtOutTime.setBounds(130, 100, 150, 30);
        add(txtOutTime);
```



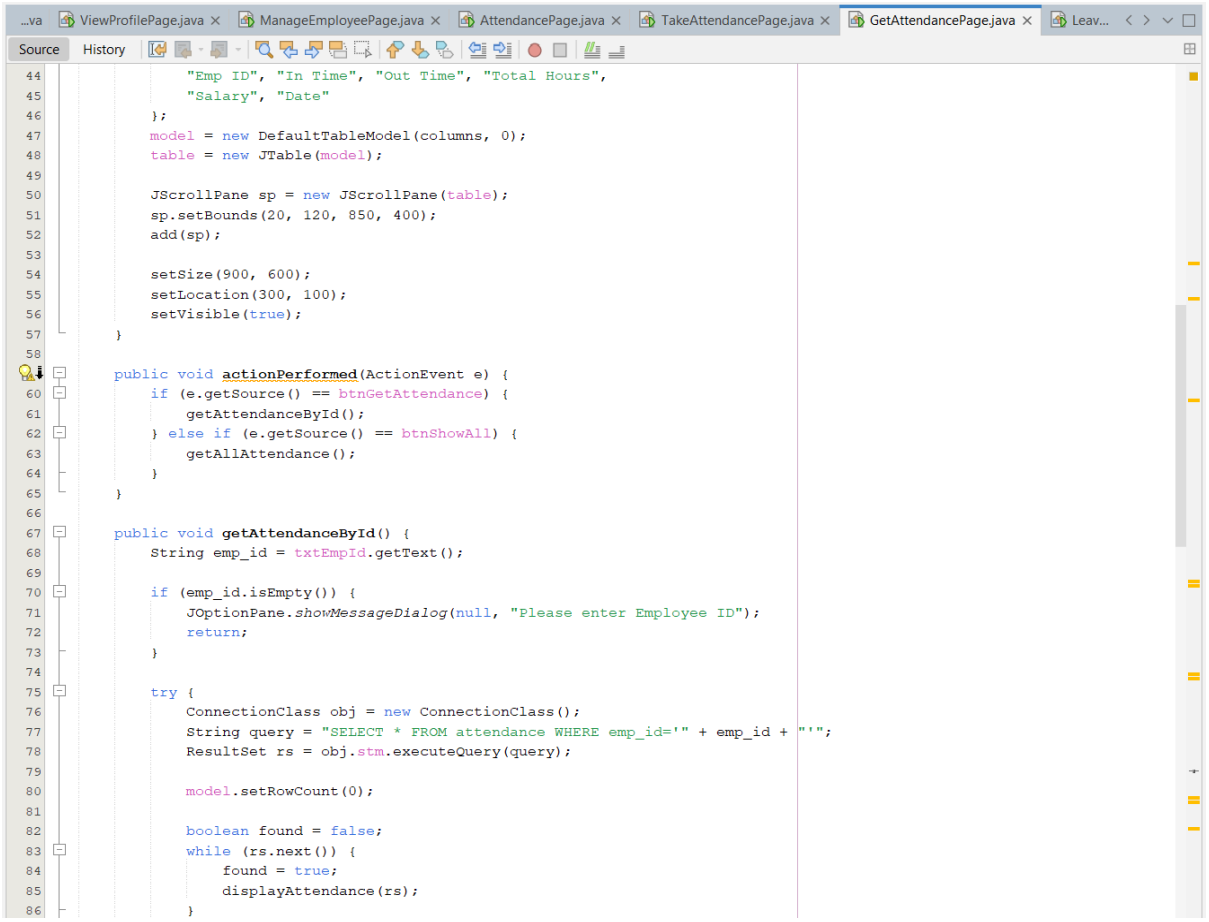
```java
        btnSave = new JButton("Save Attendance");
        btnSave.setBounds(20, 160, 150, 40);
        btnSave.setBackground(Color.BLACK);
        btnSave.setForeground(Color.WHITE);
        btnSave.addActionListener(this);
        add(btnSave);

        setSize(330, 270);
        setLocation(400, 200);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == btnSave) {
            String emp_id = txtEmpId.getText();
            String inTime = txtInTime.getText();
            String outTime = txtOutTime.getText();

            try {
                ConnectionClass obj = new ConnectionClass();
                String checkQuery = "SELECT * FROM employees WHERE emp_id='" + emp_id + "'";
                ResultSet rs = obj.stm.executeQuery(checkQuery);
                if (!rs.next()) {
                    JOptionPane.showMessageDialog(null, "Employee ID does not exist.");
                    return;
                }

                DateTimeFormatter dtf = DateTimeFormatter.ofPattern("HH:mm");
                LocalTime in = LocalTime.parse(inTime, dtf);
                LocalTime out = LocalTime.parse(outTime, dtf);

                long totalMinutes = ChronoUnit.MINUTES.between(in, out);
                double totalHours = totalMinutes / 60.0;
                double salary = totalHours * 500;

                String query = "INSERT INTO attendance (emp_id, in_time, out_time, total_hours, salary, date) " +
                        "VALUES ('" + emp_id + "', '" + inTime + "', '" + outTime + "', '" +
                        totalHours + "', '" + salary + "', CURRENT_DATE)";
                obj.stm.executeUpdate(query);

                JOptionPane.showMessageDialog(null, "Attendance Saved Successfully!");

                txtEmpId.setText("");
```



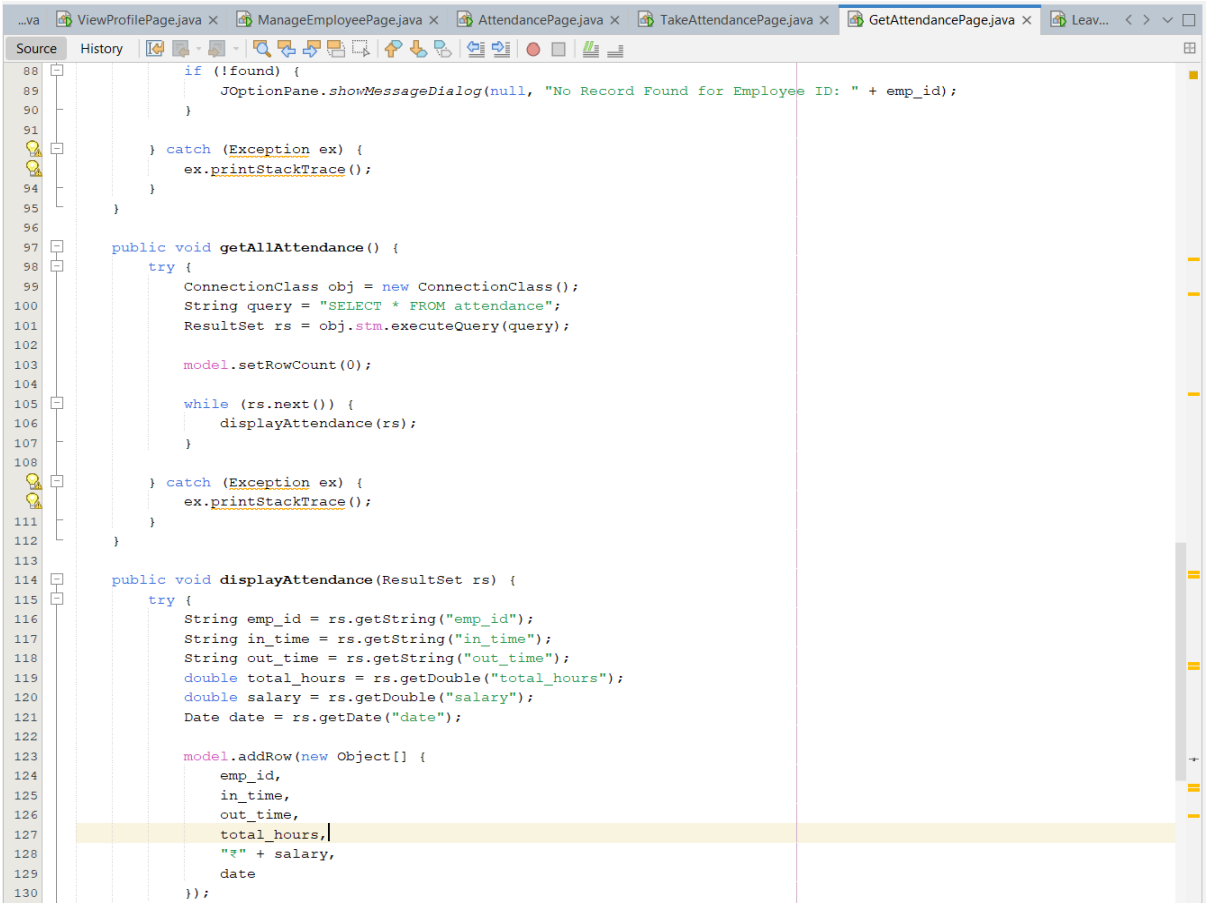
```
                    LocalTime out = LocalTime.parse(outTime, dtf);

                    long totalMinutes = ChronoUnit.MINUTES.between(in, out);
                    double totalHours = totalMinutes / 60.0;
                    double salary = totalHours * 500;

                    String query = "INSERT INTO attendance (emp_id, in_time, out_time, total_hours, salary, date) " +
                            "VALUES ('" + emp_id + "', '" + inTime + "', '" + outTime + "', '" +
                            totalHours + "', '" + salary + "', CURRENT_DATE)";
                    obj.stm.executeUpdate(query);

                    JOptionPane.showMessageDialog(null, "Attendance Saved Successfully!");

                    txtEmpId.setText("");
                    txtInTime.setText("");
                    txtOutTime.setText("");

                } catch (Exception ex) {
                    ex.printStackTrace();
                    JOptionPane.showMessageDialog(null, "Something went wrong.");
                }
            }
        }

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



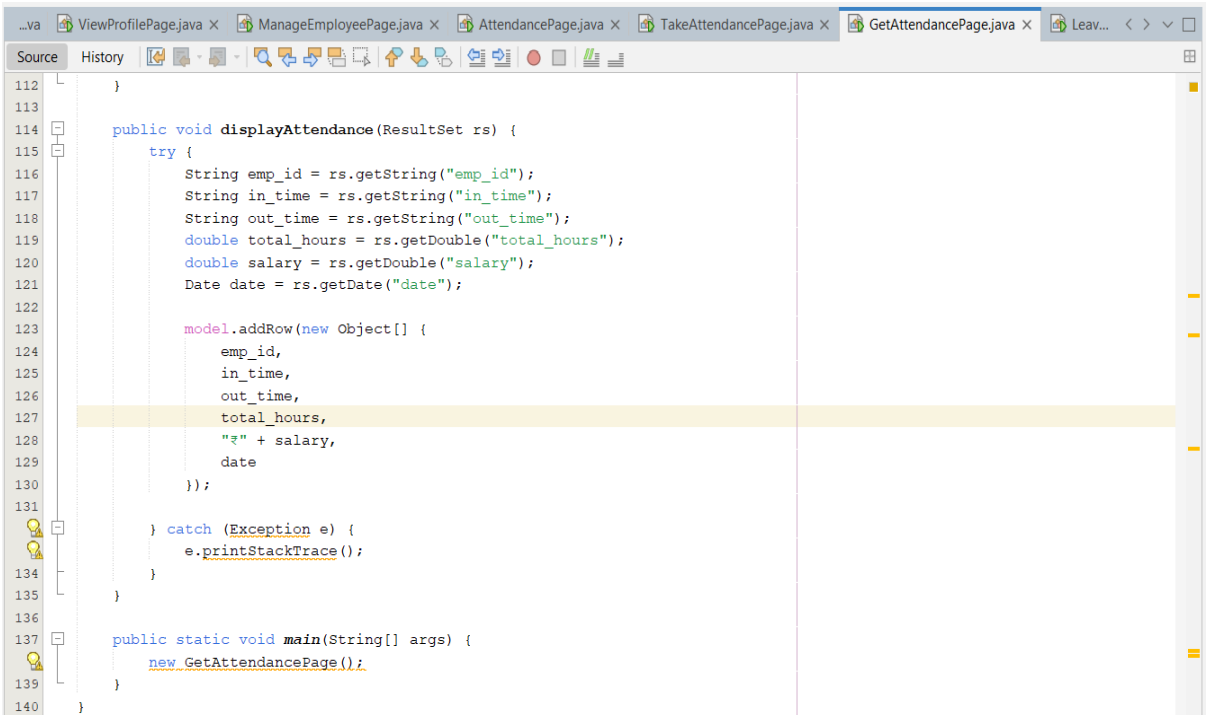
```
package Employee_Management;

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

public class GetAttendancePage extends JFrame implements ActionListener {

    JTextField txtEmpId;
    JButton btnGetAttendance, btnShowAll;
    JTable table;
    DefaultTableModel model;

    GetAttendancePage() {
        setTitle("Attendance List");
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        JLabel lblEmpId = new JLabel("Enter Employee ID:");
        lblEmpId.setBounds(20, 20, 150, 30);
        add(lblEmpId);

        txtEmpId = new JTextField();
        txtEmpId.setBounds(170, 20, 150, 30);
        add(txtEmpId);

        btnGetAttendance = new JButton("Get Attendance");
        btnGetAttendance.setBounds(20, 60, 150, 40);
        btnGetAttendance.setBackground(Color.BLACK);
        btnGetAttendance.setForeground(Color.WHITE);
        btnGetAttendance.addActionListener(this);
        add(btnGetAttendance);

        btnShowAll = new JButton("Show All");
        btnShowAll.setBounds(200, 60, 150, 40);
        btnShowAll.setBackground(Color.BLACK);
        btnShowAll.setForeground(Color.WHITE);
        btnShowAll.addActionListener(this);
        add(btnShowAll);

        String[] columns = {
```



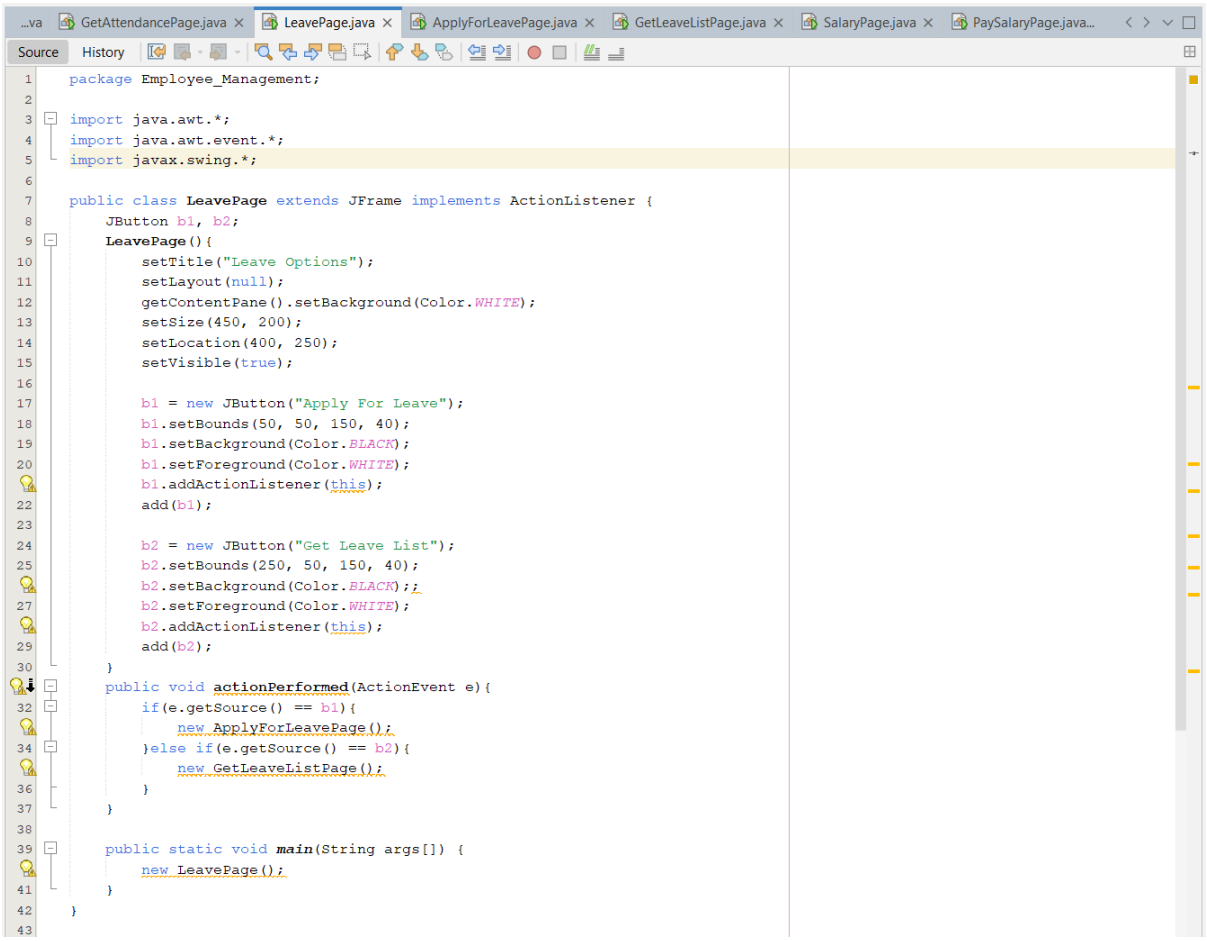
```
            "Emp ID", "In Time", "Out Time", "Total Hours",
            "Salary", "Date"
        };
        model = new DefaultTableModel(columns, 0);
        table = new JTable(model);

        JScrollPane sp = new JScrollPane(table);
        sp.setBounds(20, 120, 850, 400);
        add(sp);

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

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == btnGetAttendance) {
            getAttendanceById();
        } else if (e.getSource() == btnShowAll) {
            getAllAttendance();
        }
    }

    public void getAttendanceById() {
        String emp_id = txtEmpId.getText();

        if (emp_id.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please enter Employee ID");
            return;
        }

        try {
            ConnectionClass obj = new ConnectionClass();
            String query = "SELECT * FROM attendance WHERE emp_id='" + emp_id + "'";
            ResultSet rs = obj.stm.executeQuery(query);

            model.setRowCount(0);

            boolean found = false;
            while (rs.next()) {
                found = true;
                displayAttendance(rs);
            }
```



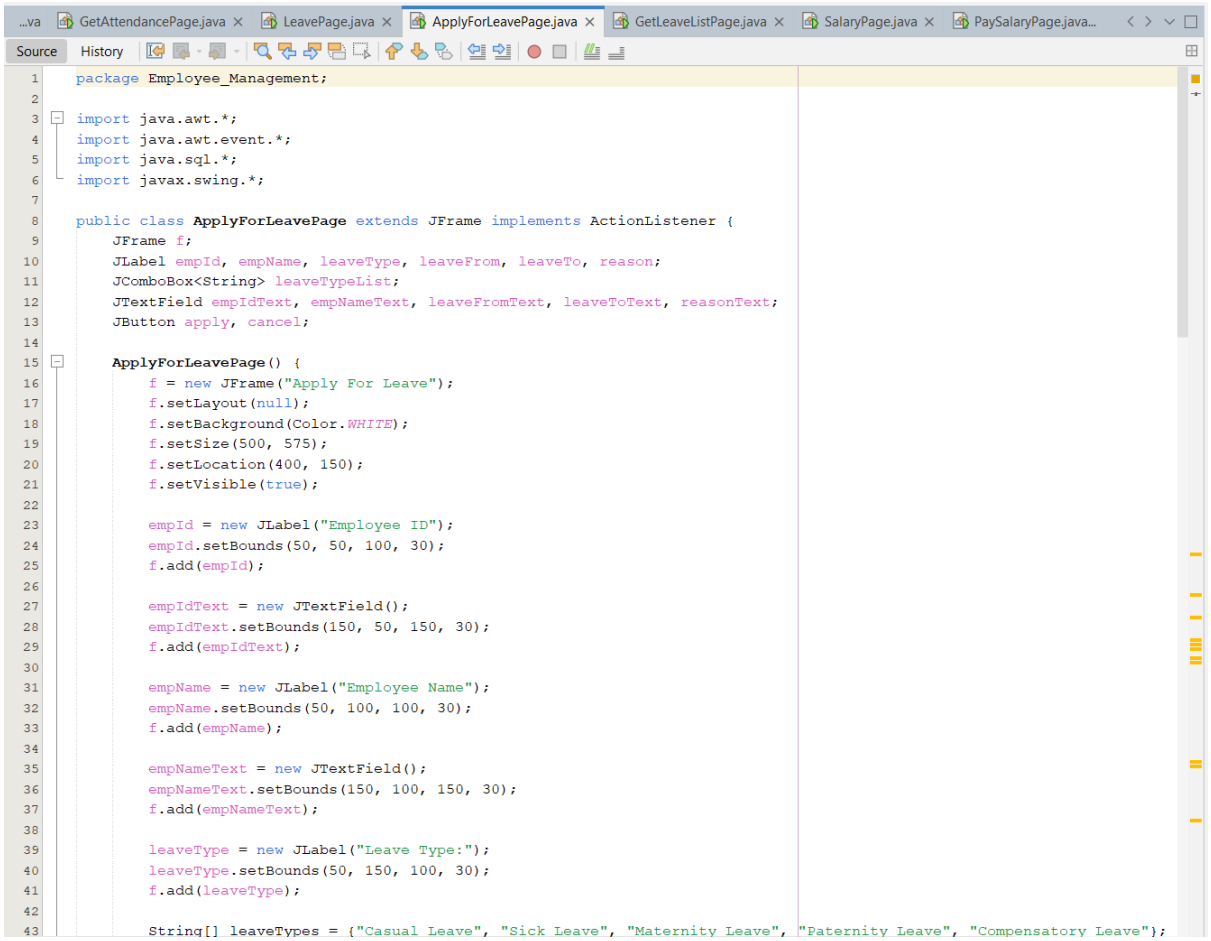
```
            if (!found) {
                JOptionPane.showMessageDialog(null, "No Record Found for Employee ID: " + emp_id);
            }

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

    public void getAllAttendance() {
        try {
            ConnectionClass obj = new ConnectionClass();
            String query = "SELECT * FROM attendance";
            ResultSet rs = obj.stm.executeQuery(query);

            model.setRowCount(0);

            while (rs.next()) {
                displayAttendance(rs);
            }

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

    public void displayAttendance(ResultSet rs) {
        try {
            String emp_id = rs.getString("emp_id");
            String in_time = rs.getString("in_time");
            String out_time = rs.getString("out_time");
            double total_hours = rs.getDouble("total_hours");
            double salary = rs.getDouble("salary");
            Date date = rs.getDate("date");

            model.addRow(new Object[] {
                emp_id,
                in_time,
                out_time,
                total_hours,
                "₹" + salary,
                date
            });
```



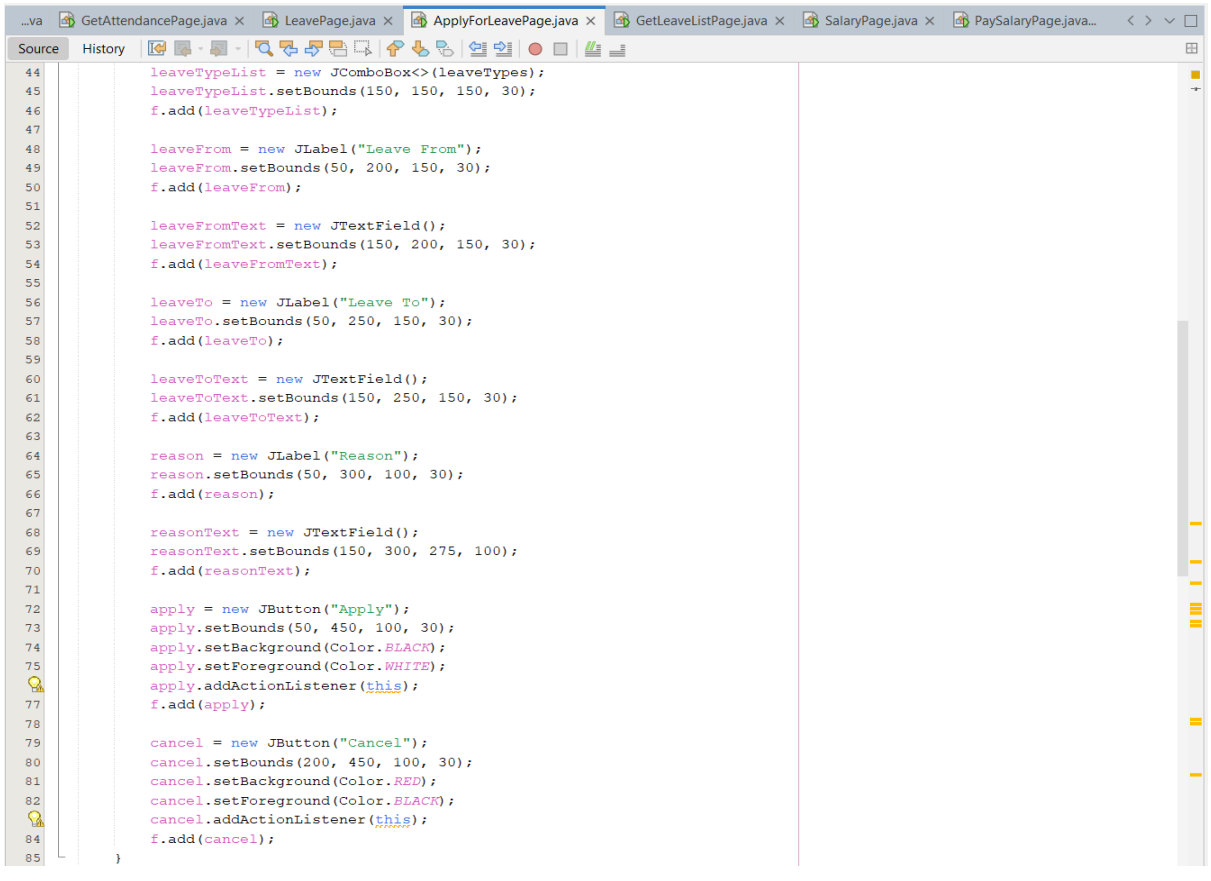
```
...va  ViewProfilePage.java ×  ManageEmployeePage.java ×  AttendancePage.java ×  TakeAttendancePage.java ×  GetAttendancePage.java ×  Leav...
Source  History
112     }
113
114     public void displayAttendance(ResultSet rs) {
115         try {
116             String emp_id = rs.getString("emp_id");
117             String in_time = rs.getString("in_time");
118             String out_time = rs.getString("out_time");
119             double total_hours = rs.getDouble("total_hours");
120             double salary = rs.getDouble("salary");
121             Date date = rs.getDate("date");
122
123             model.addRow(new Object[] {
124                 emp_id,
125                 in_time,
126                 out_time,
127                 total_hours,
128                 "₹" + salary,
129                 date
130             });
131
132         } catch (Exception e) {
133             e.printStackTrace();
134         }
135     }
136
137     public static void main(String[] args) {
138         new GetAttendancePage();
139     }
140 }
```



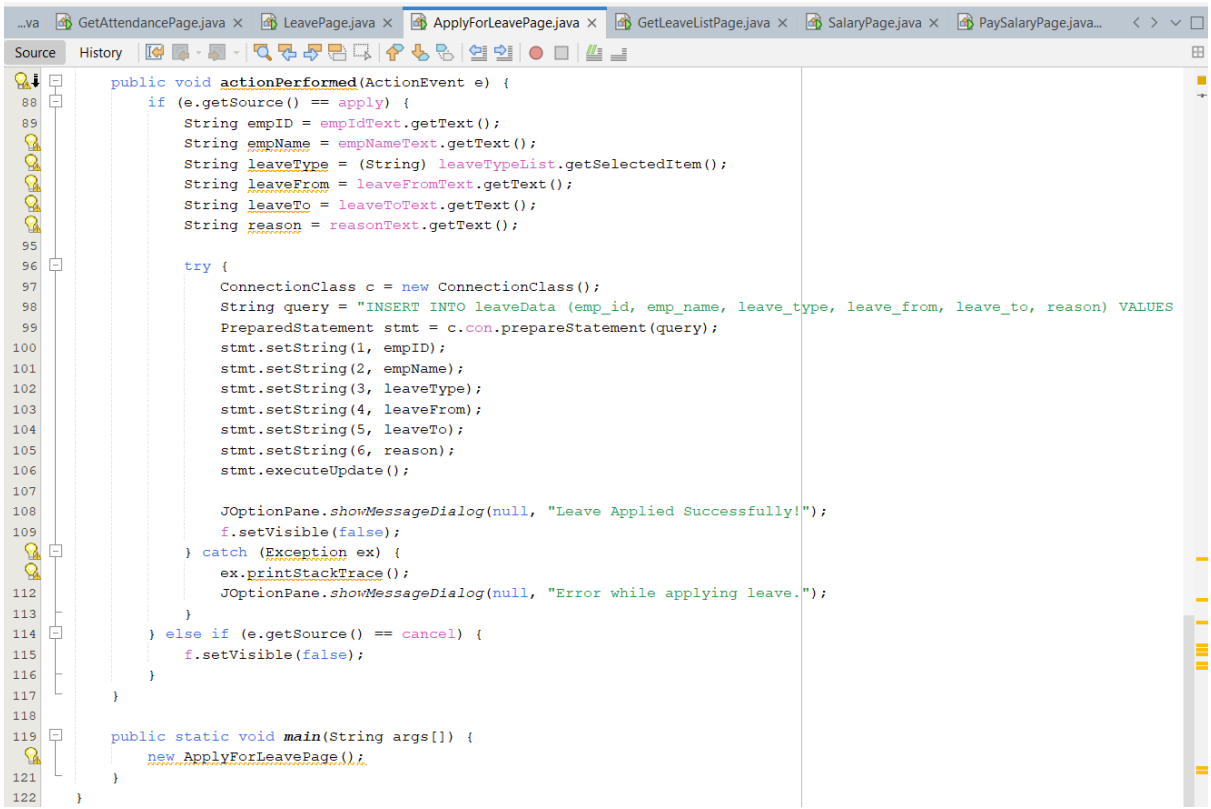
```
...va  GetAttendancePage.java ×  LeavePage.java ×  ApplyForLeavePage.java ×  GetLeaveListPage.java ×  SalaryPage.java ×  PaySalaryPage.java...
Source  History
1   package Employee_Management;
2
3   import java.awt.*;
4   import java.awt.event.*;
5   import javax.swing.*;
6
7   public class LeavePage extends JFrame implements ActionListener {
8       JButton b1, b2;
9       LeavePage(){
10          setTitle("Leave Options");
11          setLayout(null);
12          getContentPane().setBackground(Color.WHITE);
13          setSize(450, 200);
14          setLocation(400, 250);
15          setVisible(true);
16
17          b1 = new JButton("Apply For Leave");
18          b1.setBounds(50, 50, 150, 40);
19          b1.setBackground(Color.BLACK);
20          b1.setForeground(Color.WHITE);
21          b1.addActionListener(this);
22          add(b1);
23
24          b2 = new JButton("Get Leave List");
25          b2.setBounds(250, 50, 150, 40);
26          b2.setBackground(Color.BLACK);;
27          b2.setForeground(Color.WHITE);
28          b2.addActionListener(this);
29          add(b2);
30      }
31      public void actionPerformed(ActionEvent e){
32          if(e.getSource() == b1){
33              new ApplyForLeavePage();
34          }else if(e.getSource() == b2){
35              new GetLeaveListPage();
36          }
37      }
38
39      public static void main(String args[]) {
40          new LeavePage();
41      }
42  }
43
```



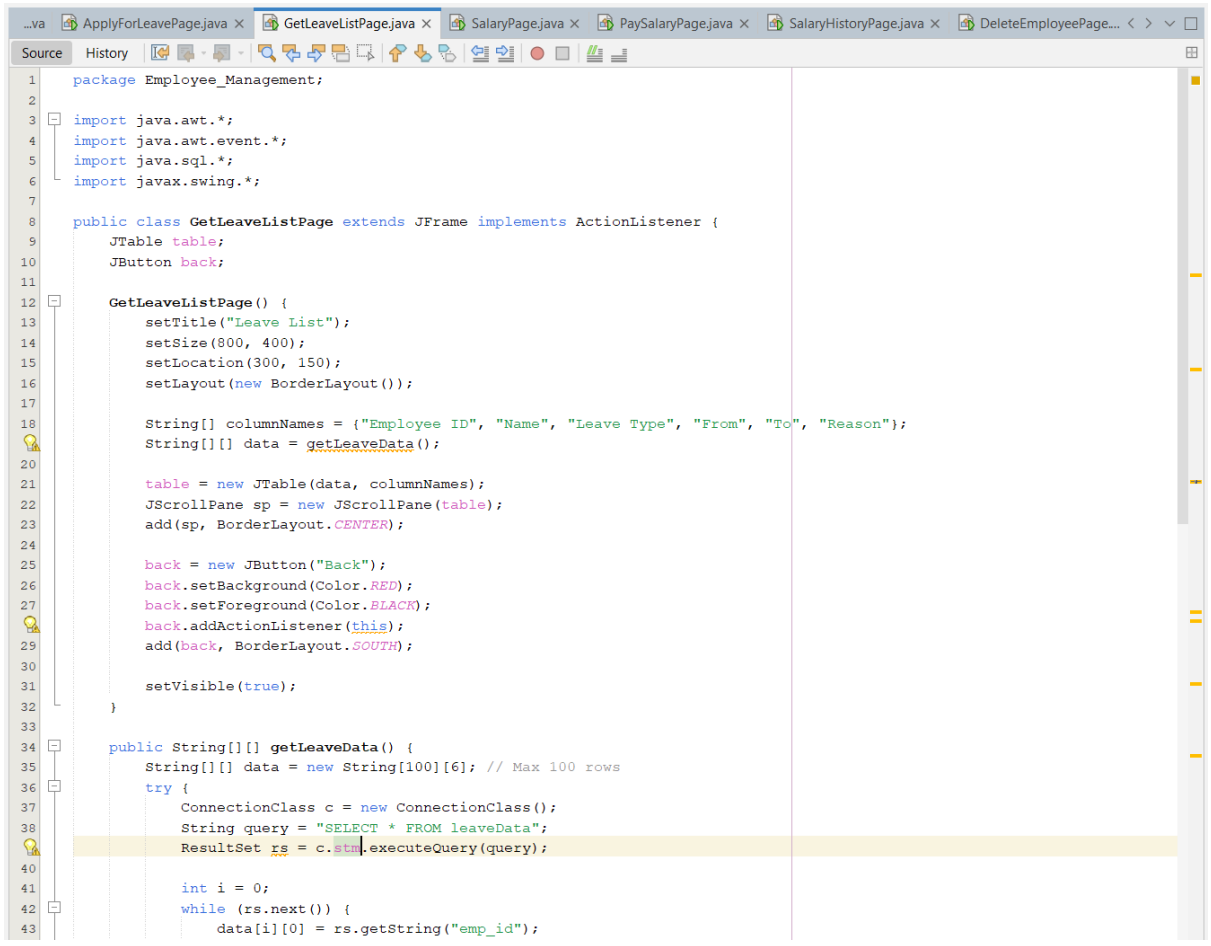
```
...va  GetAttendancePage.java  LeavePage.java  ApplyForLeavePage.java  GetLeaveListPage.java  SalaryPage.java  PaySalaryPage.java...
Source  History
1  package Employee_Management;
2
3  import java.awt.*;
4  import java.awt.event.*;
5  import java.sql.*;
6  import javax.swing.*;
7
8  public class ApplyForLeavePage extends JFrame implements ActionListener {
9      JFrame f;
10     JLabel empId, empName, leaveType, leaveFrom, leaveTo, reason;
11     JComboBox<String> leaveTypeList;
12     JTextField empIdText, empNameText, leaveFromText, leaveToText, reasonText;
13     JButton apply, cancel;
14
15     ApplyForLeavePage() {
16         f = new JFrame("Apply For Leave");
17         f.setLayout(null);
18         f.setBackground(Color.WHITE);
19         f.setSize(500, 575);
20         f.setLocation(400, 150);
21         f.setVisible(true);
22
23         empId = new JLabel("Employee ID");
24         empId.setBounds(50, 50, 100, 30);
25         f.add(empId);
26
27         empIdText = new JTextField();
28         empIdText.setBounds(150, 50, 150, 30);
29         f.add(empIdText);
30
31         empName = new JLabel("Employee Name");
32         empName.setBounds(50, 100, 100, 30);
33         f.add(empName);
34
35         empNameText = new JTextField();
36         empNameText.setBounds(150, 100, 150, 30);
37         f.add(empNameText);
38
39         leaveType = new JLabel("Leave Type:");
40         leaveType.setBounds(50, 150, 100, 30);
41         f.add(leaveType);
42
43         String[] leaveTypes = {"Casual Leave", "Sick Leave", "Maternity Leave", "Paternity Leave", "Compensatory Leave"};
```



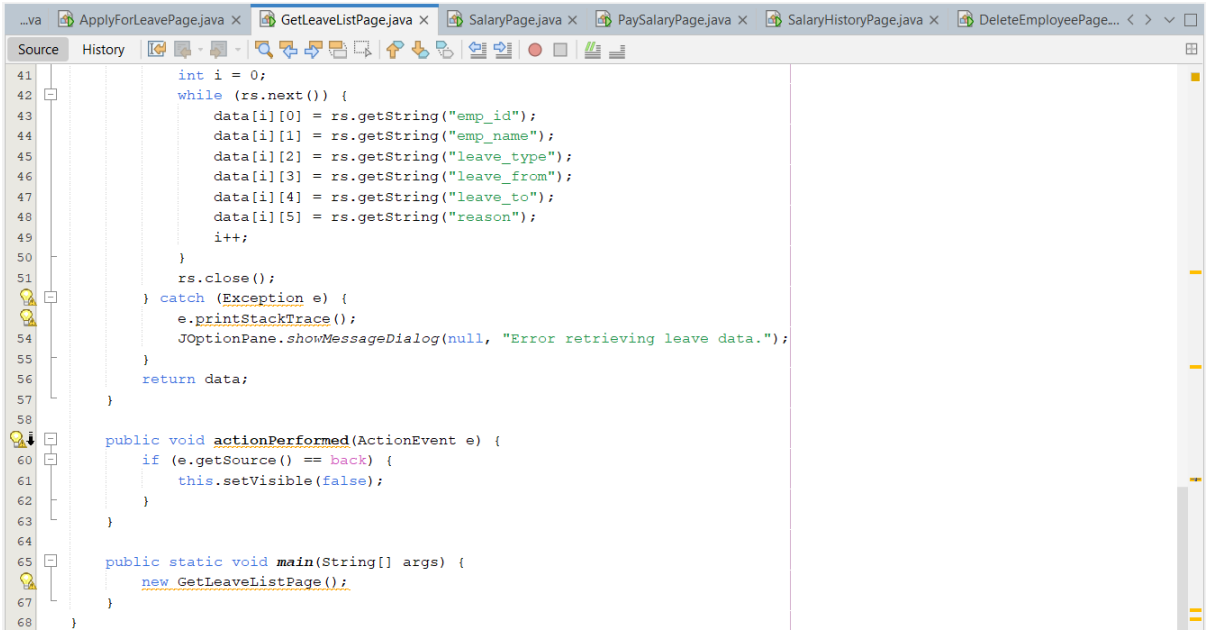
```
...va  GetAttendancePage.java  LeavePage.java  ApplyForLeavePage.java  GetLeaveListPage.java  SalaryPage.java  PaySalaryPage.java...
Source  History
44         leaveTypeList = new JComboBox<>(leaveTypes);
45         leaveTypeList.setBounds(150, 150, 150, 30);
46         f.add(leaveTypeList);
47
48         leaveFrom = new JLabel("Leave From");
49         leaveFrom.setBounds(50, 200, 150, 30);
50         f.add(leaveFrom);
51
52         leaveFromText = new JTextField();
53         leaveFromText.setBounds(150, 200, 150, 30);
54         f.add(leaveFromText);
55
56         leaveTo = new JLabel("Leave To");
57         leaveTo.setBounds(50, 250, 150, 30);
58         f.add(leaveTo);
59
60         leaveToText = new JTextField();
61         leaveToText.setBounds(150, 250, 150, 30);
62         f.add(leaveToText);
63
64         reason = new JLabel("Reason");
65         reason.setBounds(50, 300, 100, 30);
66         f.add(reason);
67
68         reasonText = new JTextField();
69         reasonText.setBounds(150, 300, 275, 100);
70         f.add(reasonText);
71
72         apply = new JButton("Apply");
73         apply.setBounds(50, 450, 100, 30);
74         apply.setBackground(Color.BLACK);
75         apply.setForeground(Color.WHITE);
76         apply.addActionListener(this);
77         f.add(apply);
78
79         cancel = new JButton("Cancel");
80         cancel.setBounds(200, 450, 100, 30);
81         cancel.setBackground(Color.RED);
82         cancel.setForeground(Color.BLACK);
83         cancel.addActionListener(this);
84         f.add(cancel);
85     }
```



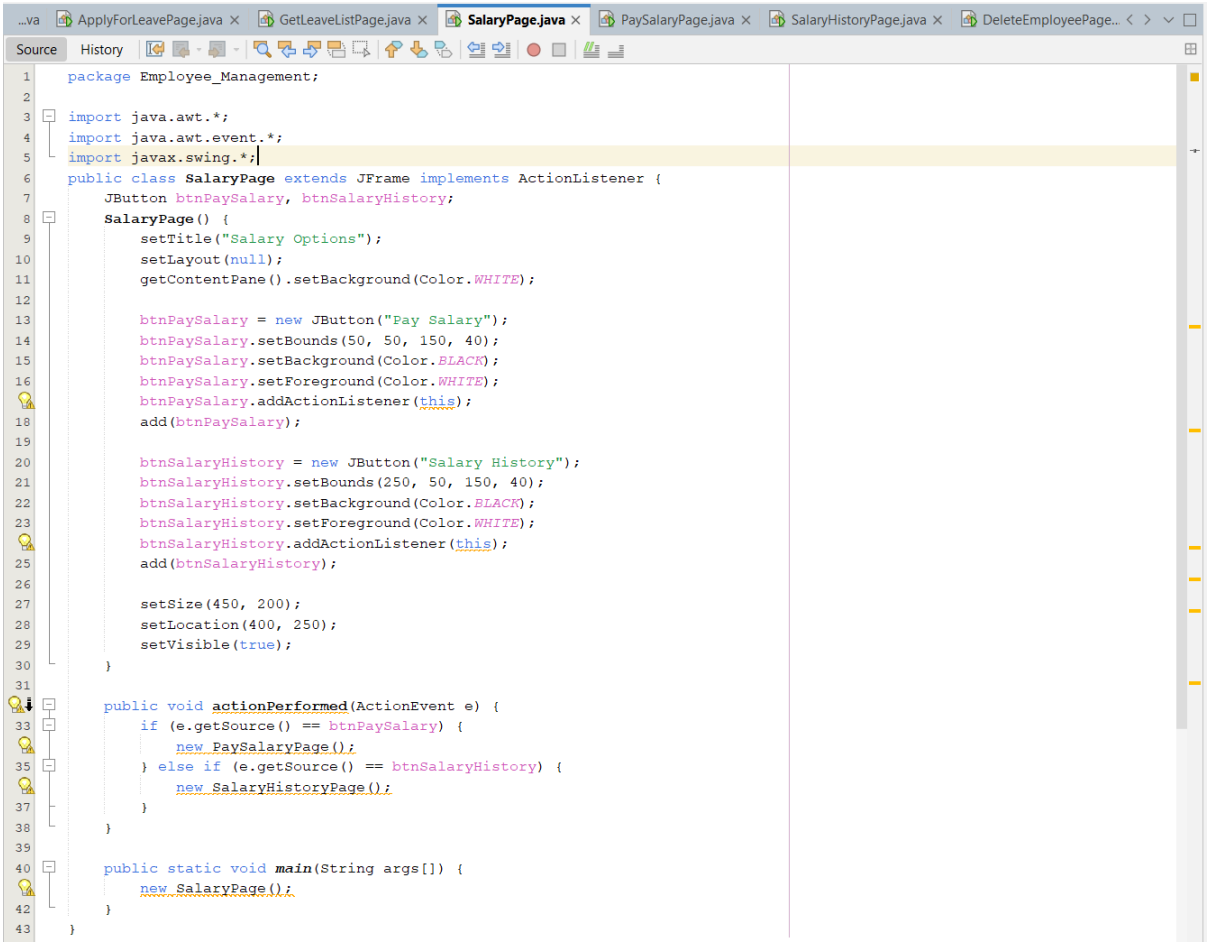
```
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == apply) {
            String empID = empIdText.getText();
            String empName = empNameText.getText();
            String leaveType = (String) leaveTypeList.getSelectedItem();
            String leaveFrom = leaveFromText.getText();
            String leaveTo = leaveToText.getText();
            String reason = reasonText.getText();

            try {
                ConnectionClass c = new ConnectionClass();
                String query = "INSERT INTO leaveData (emp_id, emp_name, leave_type, leave_from, leave_to, reason) VALUES
                PreparedStatement stmt = c.con.prepareStatement(query);
                stmt.setString(1, empID);
                stmt.setString(2, empName);
                stmt.setString(3, leaveType);
                stmt.setString(4, leaveFrom);
                stmt.setString(5, leaveTo);
                stmt.setString(6, reason);
                stmt.executeUpdate();

                JOptionPane.showMessageDialog(null, "Leave Applied Successfully!");
                f.setVisible(false);
            } catch (Exception ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(null, "Error while applying leave.");
            }
        } else if (e.getSource() == cancel) {
            f.setVisible(false);
        }
    }

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



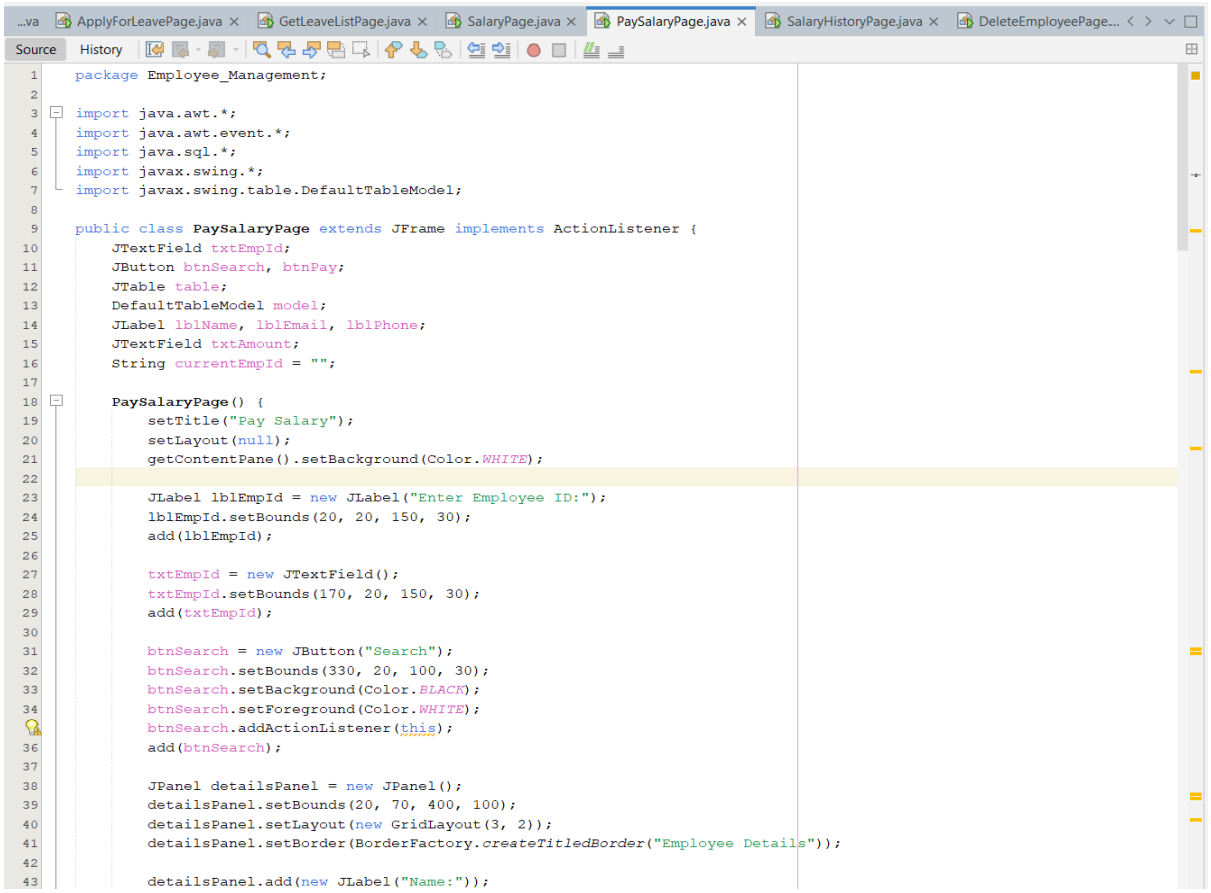
```
package Employee_Management;

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

public class GetLeaveListPage extends JFrame implements ActionListener {
    JTable table;
    JButton back;

    GetLeaveListPage() {
        setTitle("Leave List");
        setSize(800, 400);
        setLocation(300, 150);
        setLayout(new BorderLayout());

        String[] columnNames = {"Employee ID", "Name", "Leave Type", "From", "To", "Reason"};
        String[][] data = getLeaveData();

        table = new JTable(data, columnNames);
        JScrollPane sp = new JScrollPane(table);
        add(sp, BorderLayout.CENTER);

        back = new JButton("Back");
        back.setBackground(Color.RED);
        back.setForeground(Color.BLACK);
        back.addActionListener(this);
        add(back, BorderLayout.SOUTH);

        setVisible(true);
    }

    public String[][] getLeaveData() {
        String[][] data = new String[100][6]; // Max 100 rows
        try {
            ConnectionClass c = new ConnectionClass();
            String query = "SELECT * FROM leaveData";
            ResultSet rs = c.stm.executeQuery(query);

            int i = 0;
            while (rs.next()) {
                data[i][0] = rs.getString("emp_id");
```



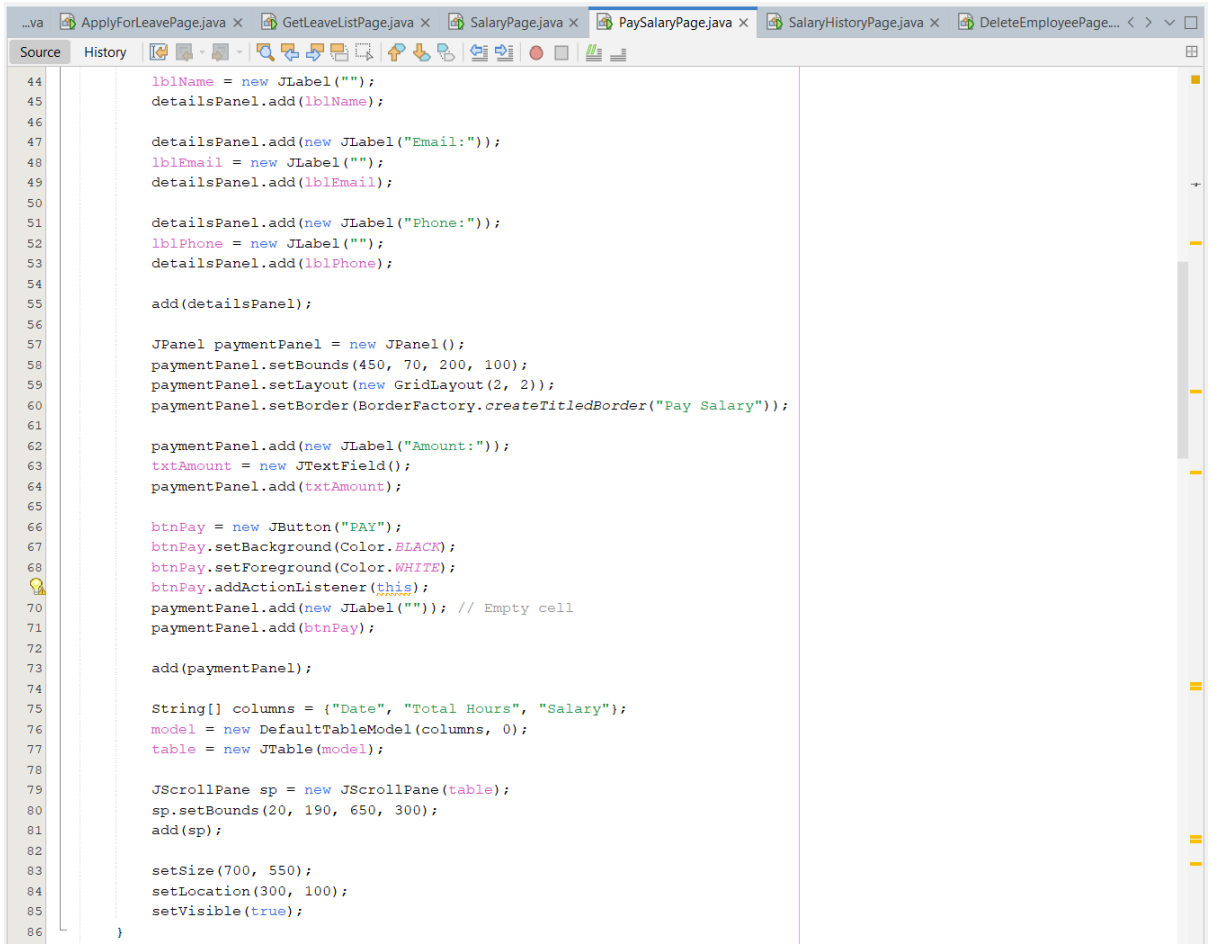
```
                int i = 0;
                while (rs.next()) {
                    data[i][0] = rs.getString("emp_id");
                    data[i][1] = rs.getString("emp_name");
                    data[i][2] = rs.getString("leave_type");
                    data[i][3] = rs.getString("leave_from");
                    data[i][4] = rs.getString("leave_to");
                    data[i][5] = rs.getString("reason");
                    i++;
                }
                rs.close();
            } catch (Exception e) {
                e.printStackTrace();
                JOptionPane.showMessageDialog(null, "Error retrieving leave data.");
            }
            return data;
        }

        public void actionPerformed(ActionEvent e) {
            if (e.getSource() == back) {
                this.setVisible(false);
            }
        }

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



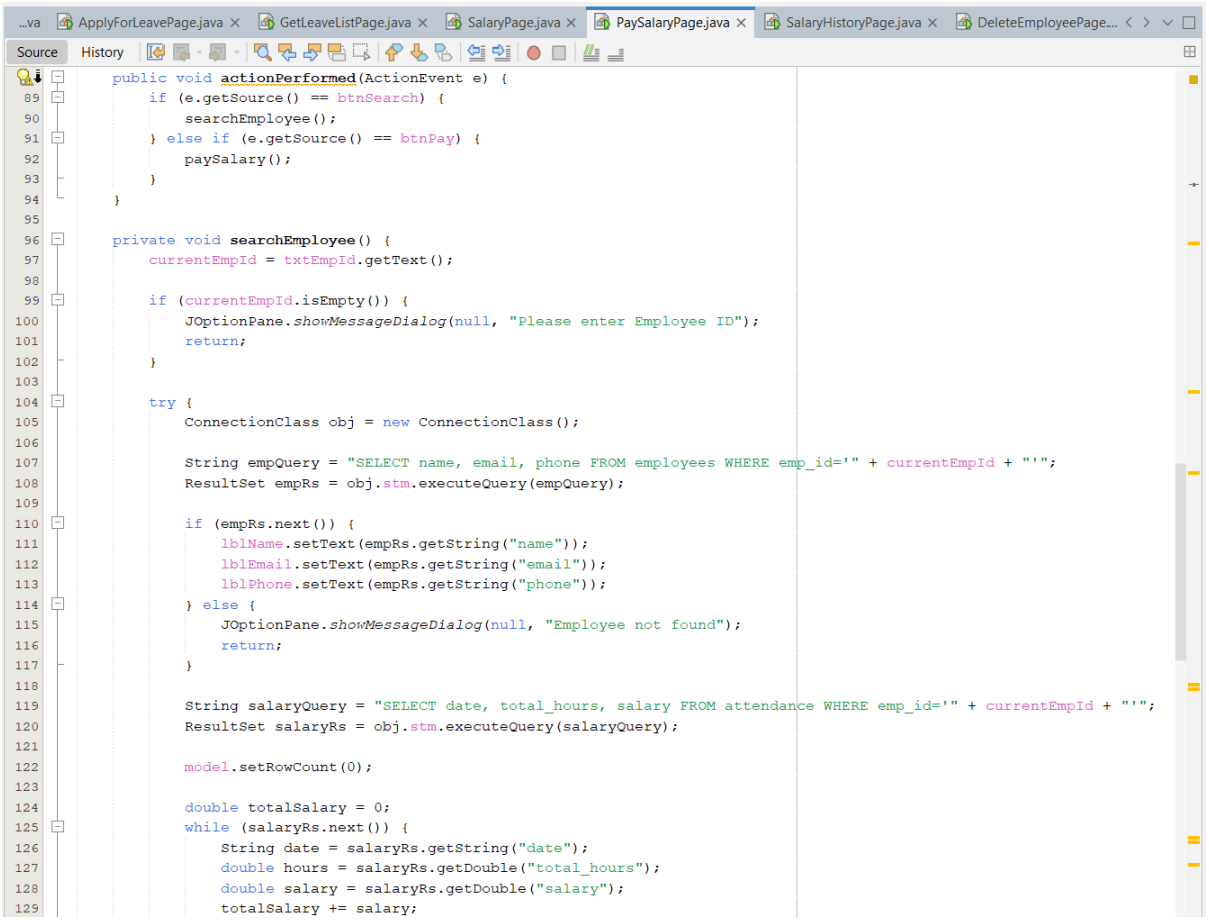
```
package Employee_Management;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class SalaryPage extends JFrame implements ActionListener {
    JButton btnPaySalary, btnSalaryHistory;
    SalaryPage() {
        setTitle("Salary Options");
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        btnPaySalary = new JButton("Pay Salary");
        btnPaySalary.setBounds(50, 50, 150, 40);
        btnPaySalary.setBackground(Color.BLACK);
        btnPaySalary.setForeground(Color.WHITE);
        btnPaySalary.addActionListener(this);
        add(btnPaySalary);

        btnSalaryHistory = new JButton("Salary History");
        btnSalaryHistory.setBounds(250, 50, 150, 40);
        btnSalaryHistory.setBackground(Color.BLACK);
        btnSalaryHistory.setForeground(Color.WHITE);
        btnSalaryHistory.addActionListener(this);
        add(btnSalaryHistory);

        setSize(450, 200);
        setLocation(400, 250);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == btnPaySalary) {
            new PaySalaryPage();
        } else if (e.getSource() == btnSalaryHistory) {
            new SalaryHistoryPage();
        }
    }

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



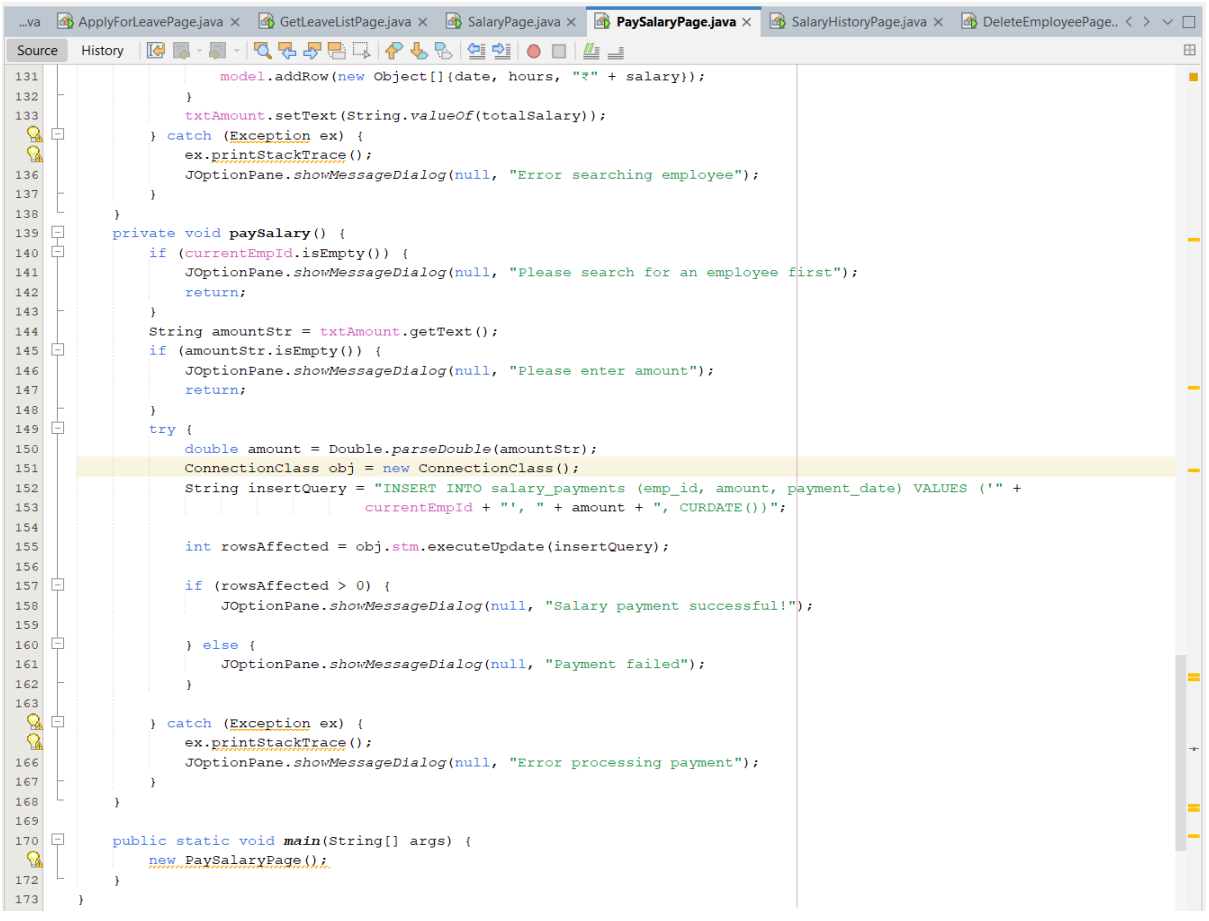
```java
package Employee_Management;

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

public class PaySalaryPage extends JFrame implements ActionListener {
    JTextField txtEmpId;
    JButton btnSearch, btnPay;
    JTable table;
    DefaultTableModel model;
    JLabel lblName, lblEmail, lblPhone;
    JTextField txtAmount;
    String currentEmpId = "";

    PaySalaryPage() {
        setTitle("Pay Salary");
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        JLabel lblEmpId = new JLabel("Enter Employee ID:");
        lblEmpId.setBounds(20, 20, 150, 30);
        add(lblEmpId);

        txtEmpId = new JTextField();
        txtEmpId.setBounds(170, 20, 150, 30);
        add(txtEmpId);

        btnSearch = new JButton("Search");
        btnSearch.setBounds(330, 20, 100, 30);
        btnSearch.setBackground(Color.BLACK);
        btnSearch.setForeground(Color.WHITE);
        btnSearch.addActionListener(this);
        add(btnSearch);

        JPanel detailsPanel = new JPanel();
        detailsPanel.setBounds(20, 70, 400, 100);
        detailsPanel.setLayout(new GridLayout(3, 2));
        detailsPanel.setBorder(BorderFactory.createTitledBorder("Employee Details"));

        detailsPanel.add(new JLabel("Name:"));
```



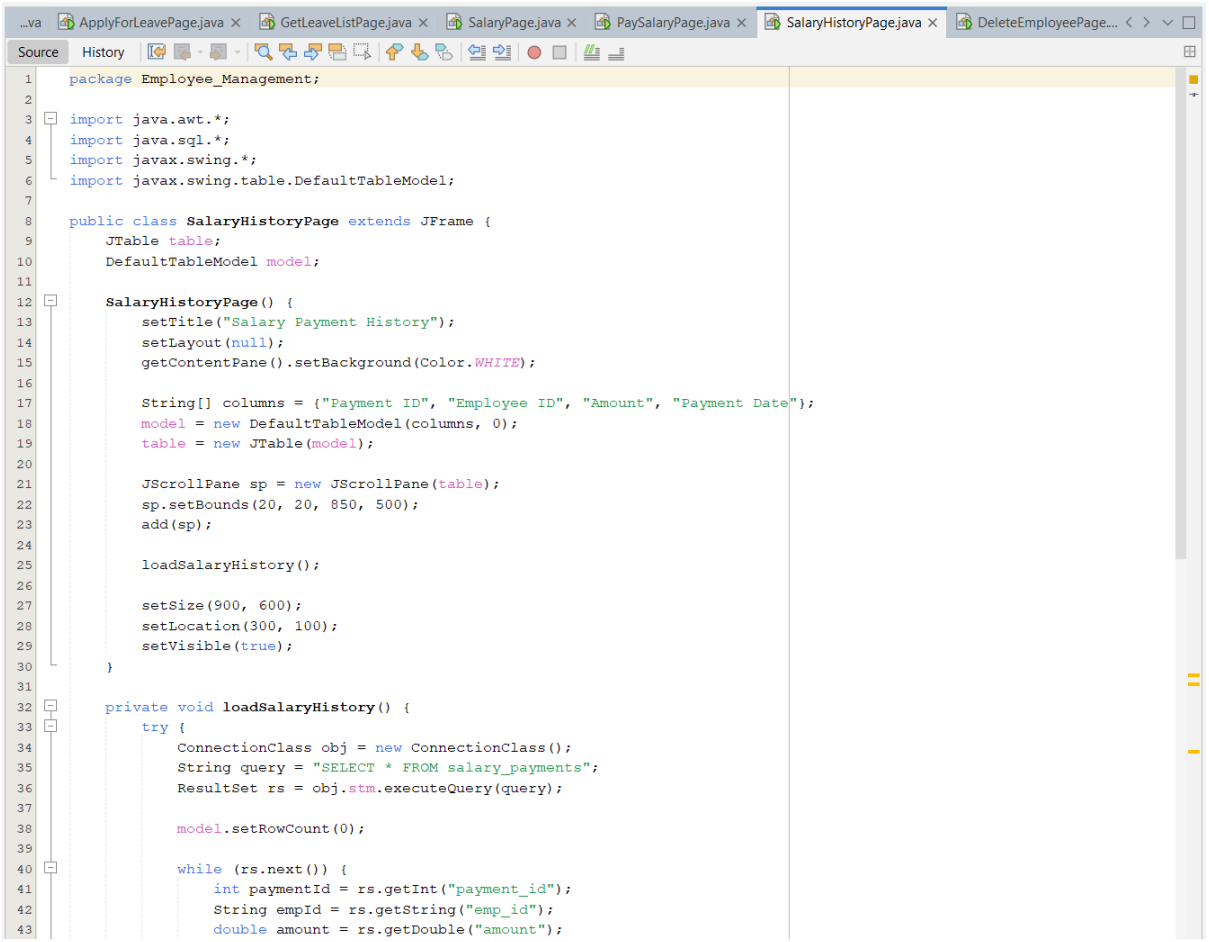
```java
        lblName = new JLabel("");
        detailsPanel.add(lblName);

        detailsPanel.add(new JLabel("Email:"));
        lblEmail = new JLabel("");
        detailsPanel.add(lblEmail);

        detailsPanel.add(new JLabel("Phone:"));
        lblPhone = new JLabel("");
        detailsPanel.add(lblPhone);

        add(detailsPanel);

        JPanel paymentPanel = new JPanel();
        paymentPanel.setBounds(450, 70, 200, 100);
        paymentPanel.setLayout(new GridLayout(2, 2));
        paymentPanel.setBorder(BorderFactory.createTitledBorder("Pay Salary"));

        paymentPanel.add(new JLabel("Amount:"));
        txtAmount = new JTextField();
        paymentPanel.add(txtAmount);

        btnPay = new JButton("PAY");
        btnPay.setBackground(Color.BLACK);
        btnPay.setForeground(Color.WHITE);
        btnPay.addActionListener(this);
        paymentPanel.add(new JLabel("")); // Empty cell
        paymentPanel.add(btnPay);

        add(paymentPanel);

        String[] columns = {"Date", "Total Hours", "Salary"};
        model = new DefaultTableModel(columns, 0);
        table = new JTable(model);

        JScrollPane sp = new JScrollPane(table);
        sp.setBounds(20, 190, 650, 300);
        add(sp);

        setSize(700, 550);
        setLocation(300, 100);
        setVisible(true);
    }
```



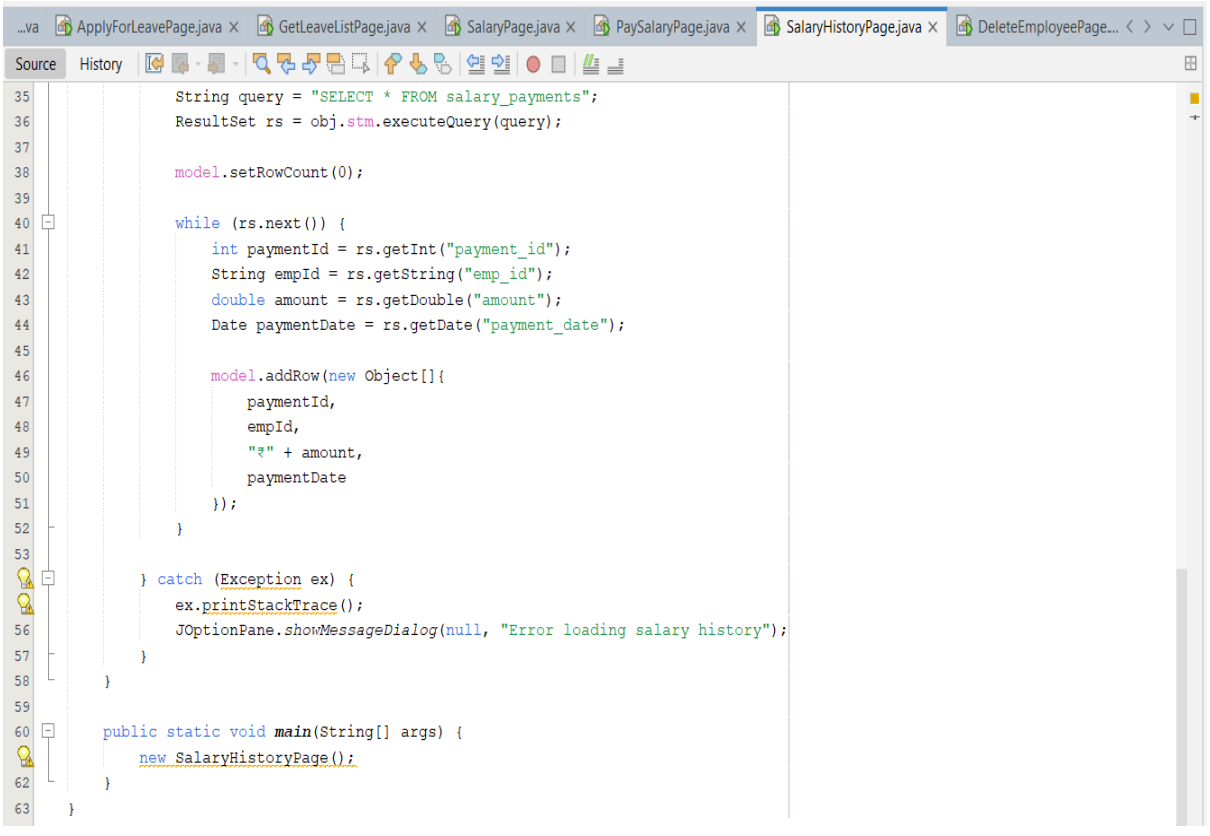
```
...va  ApplyForLeavePage.java  GetLeaveListPage.java  SalaryPage.java  PaySalaryPage.java  SalaryHistoryPage.java  DeleteEmployeePage....
Source  History

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == btnSearch) {
            searchEmployee();
        } else if (e.getSource() == btnPay) {
            paySalary();
        }
    }

    private void searchEmployee() {
        currentEmpId = txtEmpId.getText();

        if (currentEmpId.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please enter Employee ID");
            return;
        }

        try {
            ConnectionClass obj = new ConnectionClass();

            String empQuery = "SELECT name, email, phone FROM employees WHERE emp_id='" + currentEmpId + "'";
            ResultSet empRs = obj.stm.executeQuery(empQuery);

            if (empRs.next()) {
                lblName.setText(empRs.getString("name"));
                lblEmail.setText(empRs.getString("email"));
                lblPhone.setText(empRs.getString("phone"));
            } else {
                JOptionPane.showMessageDialog(null, "Employee not found");
                return;
            }

            String salaryQuery = "SELECT date, total_hours, salary FROM attendance WHERE emp_id='" + currentEmpId + "'";
            ResultSet salaryRs = obj.stm.executeQuery(salaryQuery);

            model.setRowCount(0);

            double totalSalary = 0;
            while (salaryRs.next()) {
                String date = salaryRs.getString("date");
                double hours = salaryRs.getDouble("total_hours");
                double salary = salaryRs.getDouble("salary");
                totalSalary += salary;
```



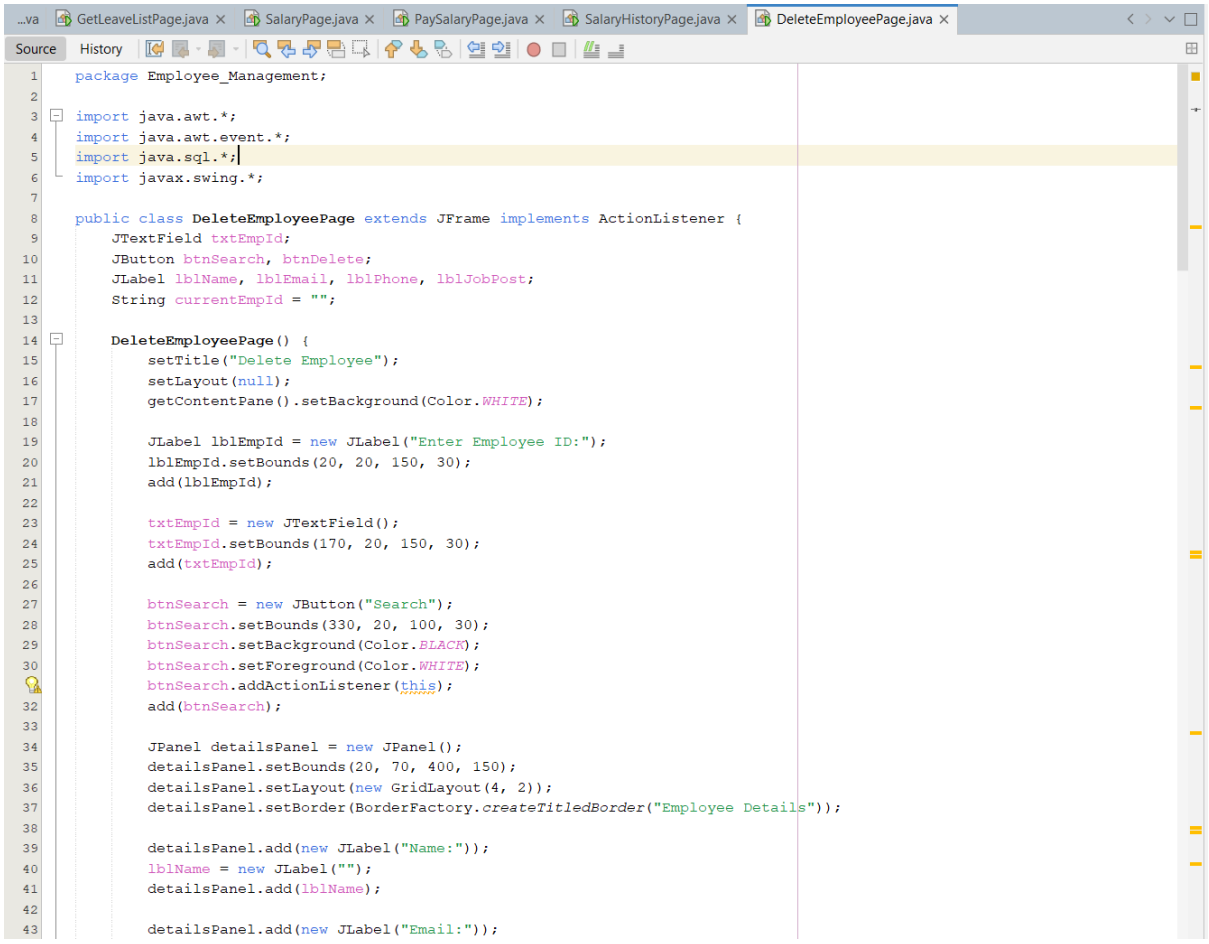
```
...va  ApplyForLeavePage.java  GetLeaveListPage.java  SalaryPage.java  PaySalaryPage.java  SalaryHistoryPage.java  DeleteEmployeePage..
Source  History

                model.addRow(new Object[]{date, hours, "₹" + salary});
            }
            txtAmount.setText(String.valueOf(totalSalary));
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error searching employee");
        }
    }
    private void paySalary() {
        if (currentEmpId.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please search for an employee first");
            return;
        }
        String amountStr = txtAmount.getText();
        if (amountStr.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Please enter amount");
            return;
        }
        try {
            double amount = Double.parseDouble(amountStr);
            ConnectionClass obj = new ConnectionClass();
            String insertQuery = "INSERT INTO salary_payments (emp_id, amount, payment_date) VALUES ('" +
                                 currentEmpId + "', " + amount + ", CURDATE())";

            int rowsAffected = obj.stm.executeUpdate(insertQuery);

            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(null, "Salary payment successful!");

            } else {
                JOptionPane.showMessageDialog(null, "Payment failed");
            }

        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error processing payment");
        }
    }

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



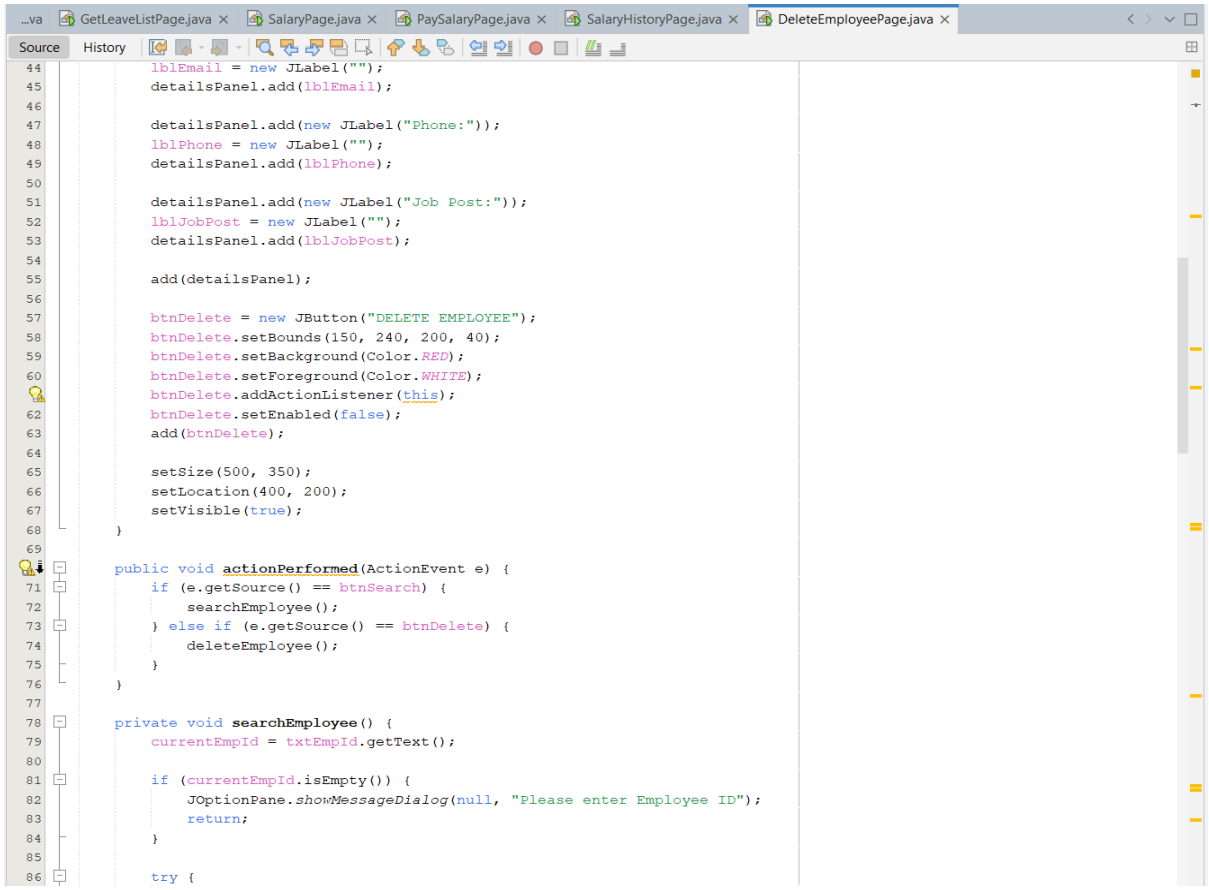
```java
package Employee_Management;

import java.awt.*;
import java.sql.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

public class SalaryHistoryPage extends JFrame {
    JTable table;
    DefaultTableModel model;

    SalaryHistoryPage() {
        setTitle("Salary Payment History");
        setLayout(null);
        getContentPane().setBackground(Color.WHITE);

        String[] columns = {"Payment ID", "Employee ID", "Amount", "Payment Date"};
        model = new DefaultTableModel(columns, 0);
        table = new JTable(model);

        JScrollPane sp = new JScrollPane(table);
        sp.setBounds(20, 20, 850, 500);
        add(sp);

        loadSalaryHistory();

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

    private void loadSalaryHistory() {
        try {
            ConnectionClass obj = new ConnectionClass();
            String query = "SELECT * FROM salary_payments";
            ResultSet rs = obj.stm.executeQuery(query);

            model.setRowCount(0);

            while (rs.next()) {
                int paymentId = rs.getInt("payment_id");
                String empId = rs.getString("emp_id");
                double amount = rs.getDouble("amount");
```



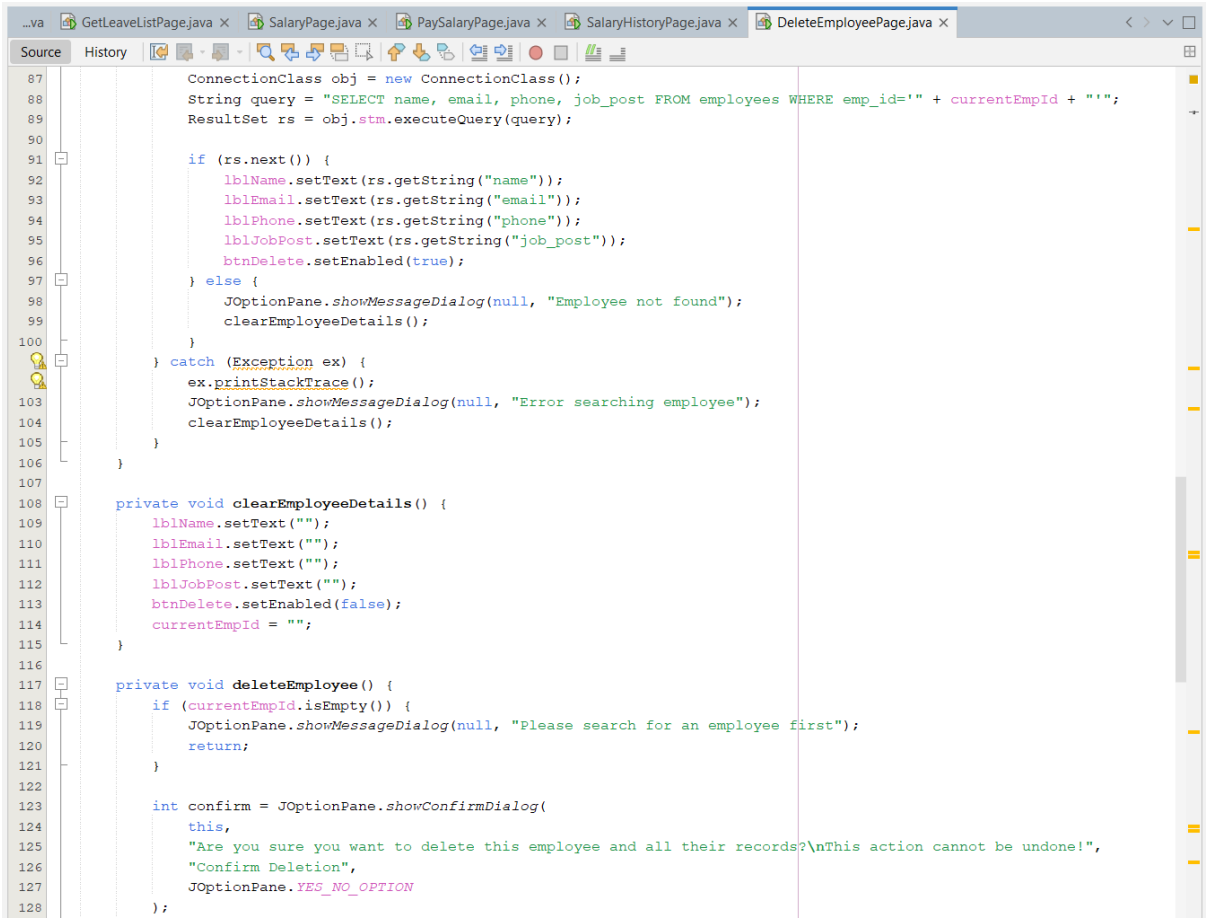
```java
                String query = "SELECT * FROM salary_payments";
                ResultSet rs = obj.stm.executeQuery(query);

                model.setRowCount(0);

                while (rs.next()) {
                    int paymentId = rs.getInt("payment_id");
                    String empId = rs.getString("emp_id");
                    double amount = rs.getDouble("amount");
                    Date paymentDate = rs.getDate("payment_date");

                    model.addRow(new Object[]{
                        paymentId,
                        empId,
                        "₹" + amount,
                        paymentDate
                    });
                }

            } catch (Exception ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(null, "Error loading salary history");
            }
        }

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



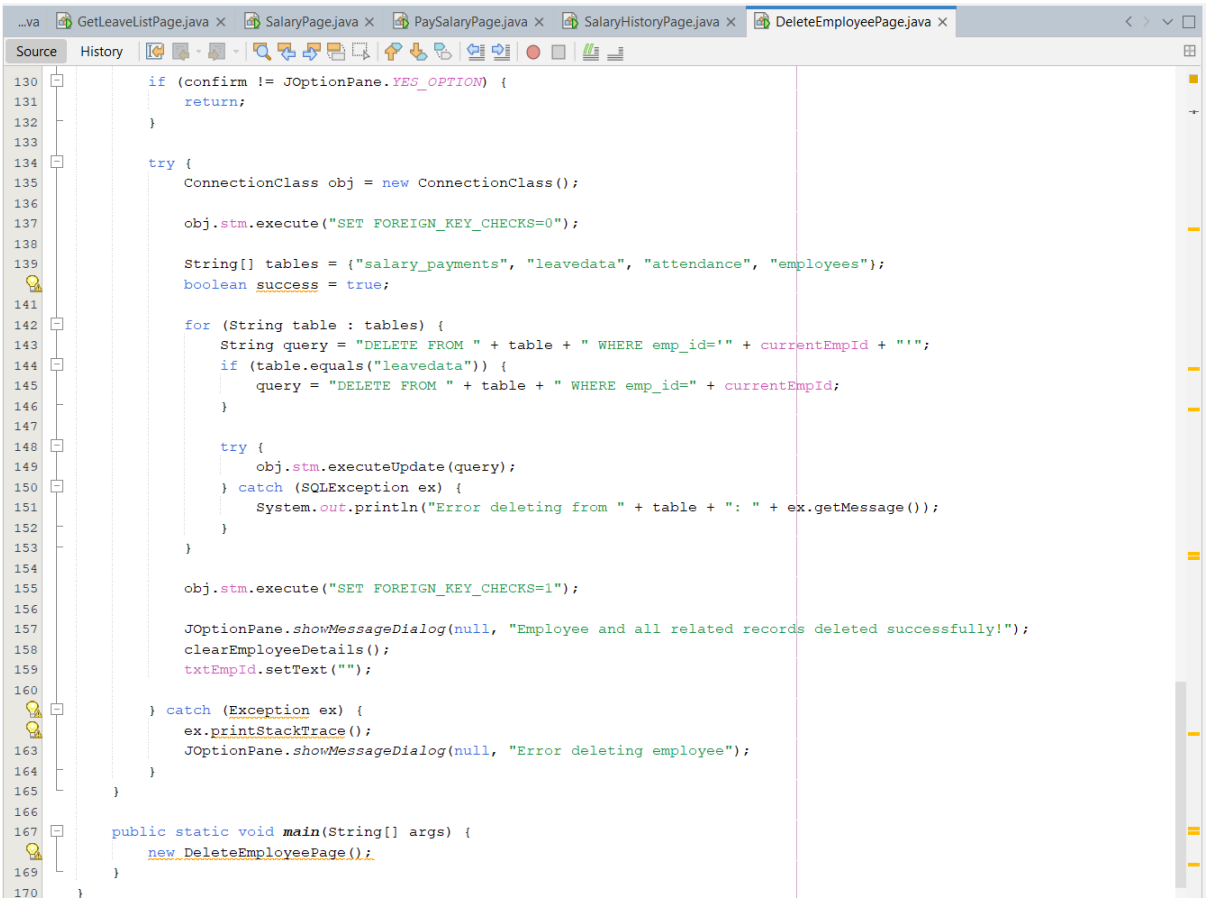
```
...va | GetLeaveListPage.java | SalaryPage.java | PaySalaryPage.java | SalaryHistoryPage.java | DeleteEmployeePage.java
Source | History

1  package Employee_Management;
2
3  import java.awt.*;
4  import java.awt.event.*;
5  import java.sql.*;
6  import javax.swing.*;
7
8  public class DeleteEmployeePage extends JFrame implements ActionListener {
9      JTextField txtEmpId;
10     JButton btnSearch, btnDelete;
11     JLabel lblName, lblEmail, lblPhone, lblJobPost;
12     String currentEmpId = "";
13
14     DeleteEmployeePage() {
15         setTitle("Delete Employee");
16         setLayout(null);
17         getContentPane().setBackground(Color.WHITE);
18
19         JLabel lblEmpId = new JLabel("Enter Employee ID:");
20         lblEmpId.setBounds(20, 20, 150, 30);
21         add(lblEmpId);
22
23         txtEmpId = new JTextField();
24         txtEmpId.setBounds(170, 20, 150, 30);
25         add(txtEmpId);
26
27         btnSearch = new JButton("Search");
28         btnSearch.setBounds(330, 20, 100, 30);
29         btnSearch.setBackground(Color.BLACK);
30         btnSearch.setForeground(Color.WHITE);
31         btnSearch.addActionListener(this);
32         add(btnSearch);
33
34         JPanel detailsPanel = new JPanel();
35         detailsPanel.setBounds(20, 70, 400, 150);
36         detailsPanel.setLayout(new GridLayout(4, 2));
37         detailsPanel.setBorder(BorderFactory.createTitledBorder("Employee Details"));
38
39         detailsPanel.add(new JLabel("Name:"));
40         lblName = new JLabel("");
41         detailsPanel.add(lblName);
42
43         detailsPanel.add(new JLabel("Email:"));
```



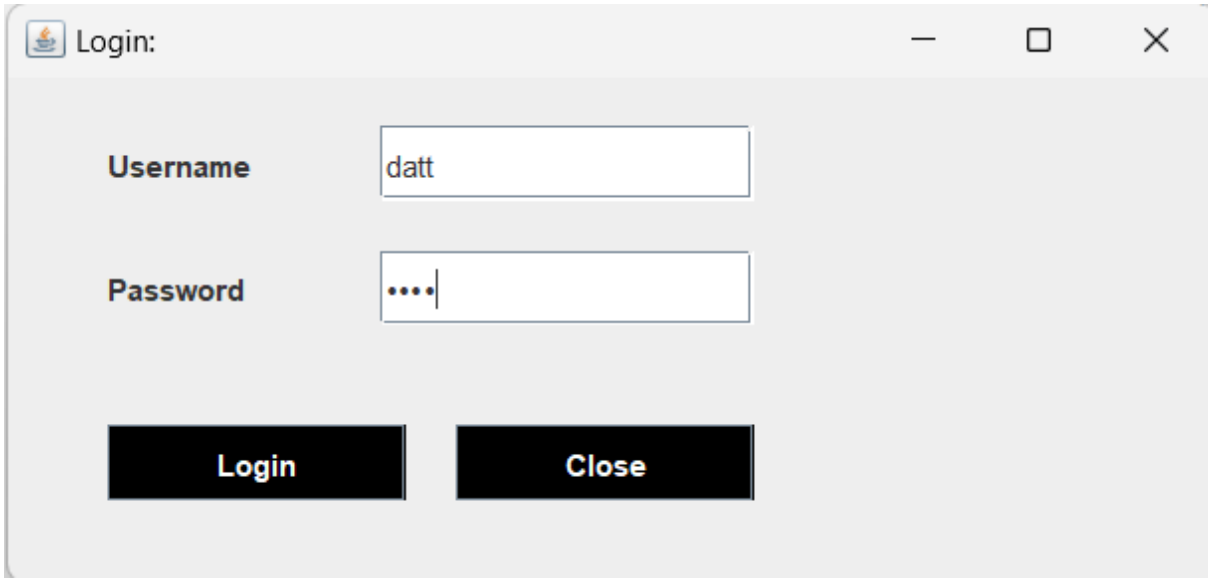
```
...va | GetLeaveListPage.java | SalaryPage.java | PaySalaryPage.java | SalaryHistoryPage.java | DeleteEmployeePage.java
Source | History

44         lblEmail = new JLabel("");
45         detailsPanel.add(lblEmail);
46
47         detailsPanel.add(new JLabel("Phone:"));
48         lblPhone = new JLabel("");
49         detailsPanel.add(lblPhone);
50
51         detailsPanel.add(new JLabel("Job Post:"));
52         lblJobPost = new JLabel("");
53         detailsPanel.add(lblJobPost);
54
55         add(detailsPanel);
56
57         btnDelete = new JButton("DELETE EMPLOYEE");
58         btnDelete.setBounds(150, 240, 200, 40);
59         btnDelete.setBackground(Color.RED);
60         btnDelete.setForeground(Color.WHITE);
61         btnDelete.addActionListener(this);
62         btnDelete.setEnabled(false);
63         add(btnDelete);
64
65         setSize(500, 350);
66         setLocation(400, 200);
67         setVisible(true);
68     }
69
70     public void actionPerformed(ActionEvent e) {
71         if (e.getSource() == btnSearch) {
72             searchEmployee();
73         } else if (e.getSource() == btnDelete) {
74             deleteEmployee();
75         }
76     }
77
78     private void searchEmployee() {
79         currentEmpId = txtEmpId.getText();
80
81         if (currentEmpId.isEmpty()) {
82             JOptionPane.showMessageDialog(null, "Please enter Employee ID");
83             return;
84         }
85
86         try {
```

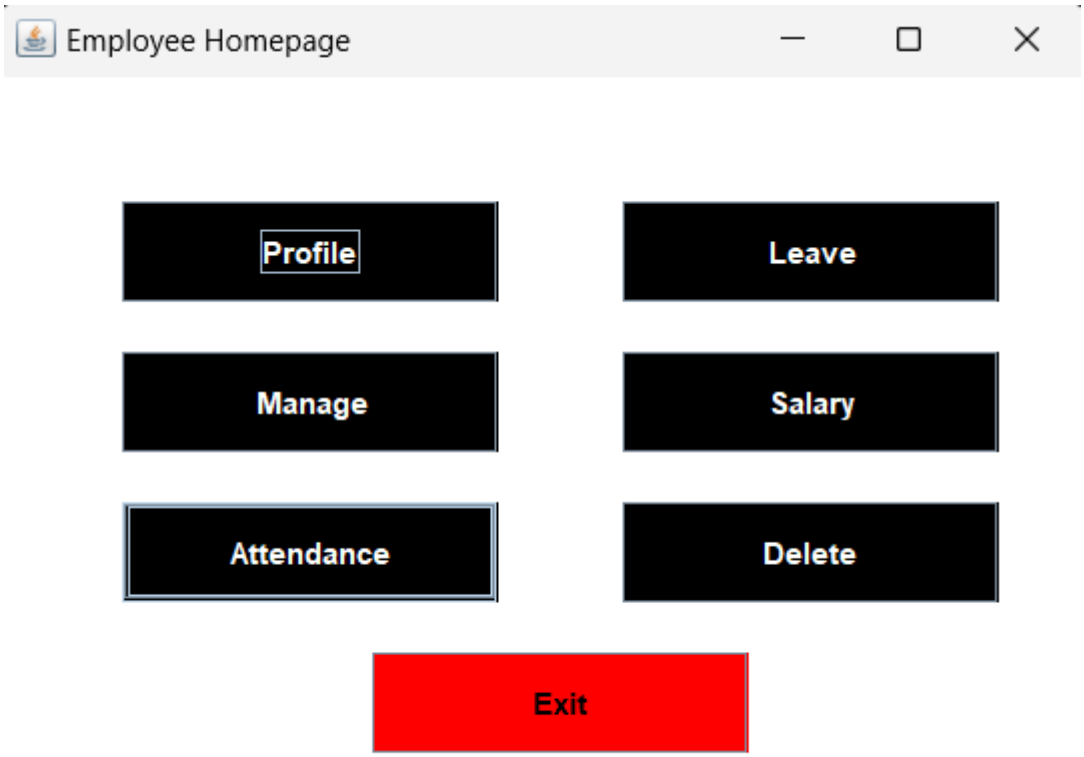



```
87              ConnectionClass obj = new ConnectionClass();
88              String query = "SELECT name, email, phone, job_post FROM employees WHERE emp_id='" + currentEmpId + "'";
89              ResultSet rs = obj.stm.executeQuery(query);
90
91              if (rs.next()) {
92                  lblName.setText(rs.getString("name"));
93                  lblEmail.setText(rs.getString("email"));
94                  lblPhone.setText(rs.getString("phone"));
95                  lblJobPost.setText(rs.getString("job_post"));
96                  btnDelete.setEnabled(true);
97              } else {
98                  JOptionPane.showMessageDialog(null, "Employee not found");
99                  clearEmployeeDetails();
100             }
101         } catch (Exception ex) {
102             ex.printStackTrace();
103             JOptionPane.showMessageDialog(null, "Error searching employee");
104             clearEmployeeDetails();
105         }
106     }
107
108     private void clearEmployeeDetails() {
109         lblName.setText("");
110         lblEmail.setText("");
111         lblPhone.setText("");
112         lblJobPost.setText("");
113         btnDelete.setEnabled(false);
114         currentEmpId = "";
115     }
116
117     private void deleteEmployee() {
118         if (currentEmpId.isEmpty()) {
119             JOptionPane.showMessageDialog(null, "Please search for an employee first");
120             return;
121         }
122
123         int confirm = JOptionPane.showConfirmDialog(
124             this,
125             "Are you sure you want to delete this employee and all their records?\nThis action cannot be undone!",
126             "Confirm Deletion",
127             JOptionPane.YES_NO_OPTION
128         );
```

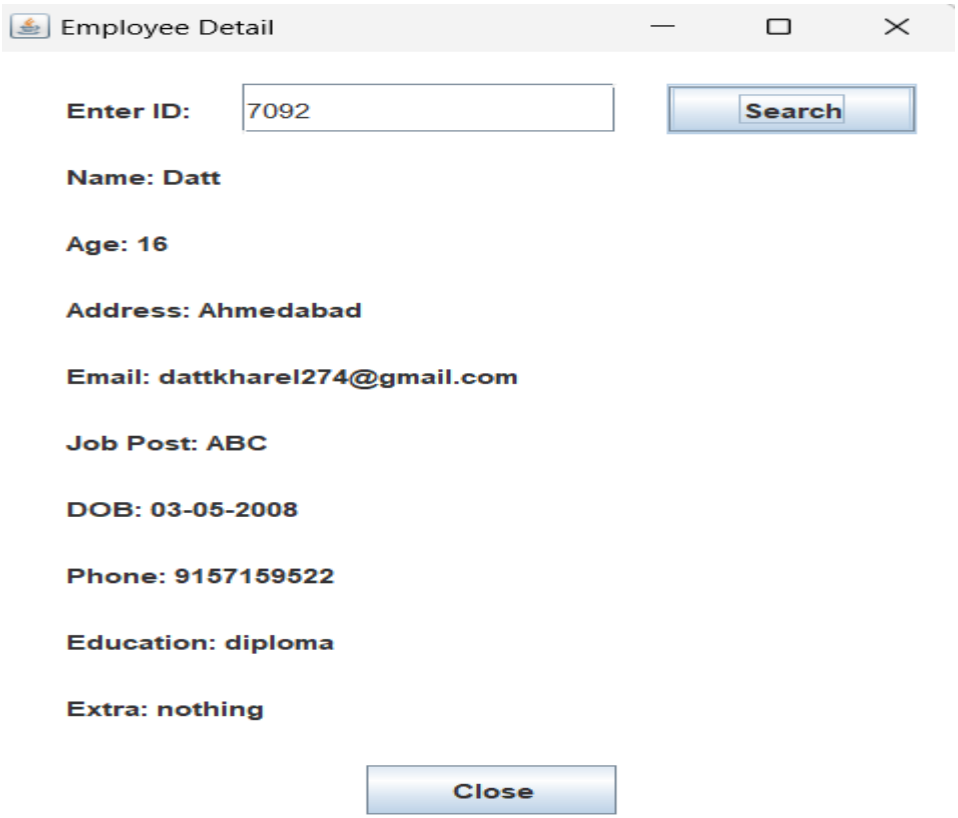


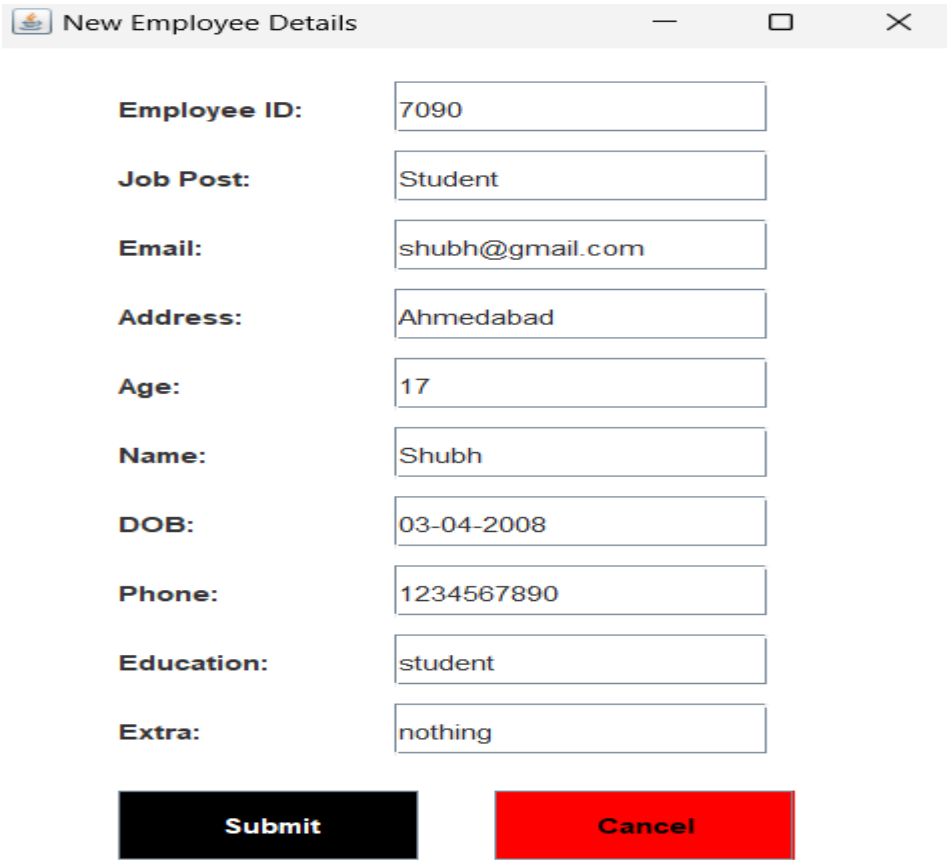

```
130         if (confirm != JOptionPane.YES_OPTION) {
131             return;
132         }
133
134         try {
135             ConnectionClass obj = new ConnectionClass();
136
137             obj.stm.execute("SET FOREIGN_KEY_CHECKS=0");
138
139             String[] tables = {"salary_payments", "leavedata", "attendance", "employees"};
140             boolean success = true;
141
142             for (String table : tables) {
143                 String query = "DELETE FROM " + table + " WHERE emp_id='" + currentEmpId + "'";
144                 if (table.equals("leavedata")) {
145                     query = "DELETE FROM " + table + " WHERE emp_id=" + currentEmpId;
146                 }
147
148                 try {
149                     obj.stm.executeUpdate(query);
150                 } catch (SQLException ex) {
151                     System.out.println("Error deleting from " + table + ": " + ex.getMessage());
152                 }
153             }
154
155             obj.stm.execute("SET FOREIGN_KEY_CHECKS=1");
156
157             JOptionPane.showMessageDialog(null, "Employee and all related records deleted successfully!");
158             clearEmployeeDetails();
159             txtEmpId.setText("");
160
161         } catch (Exception ex) {
162             ex.printStackTrace();
163             JOptionPane.showMessageDialog(null, "Error deleting employee");
164         }
165     }
166
167     public static void main(String[] args) {
168         new DeleteEmployeePage();
169     }
170 }
```

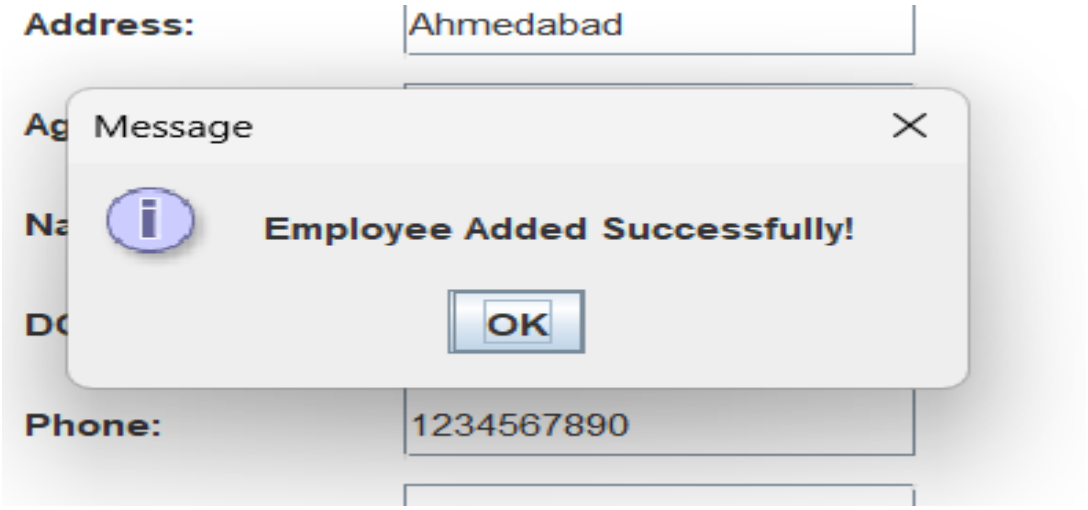

Login:
Username
datt
Password
Login
Close

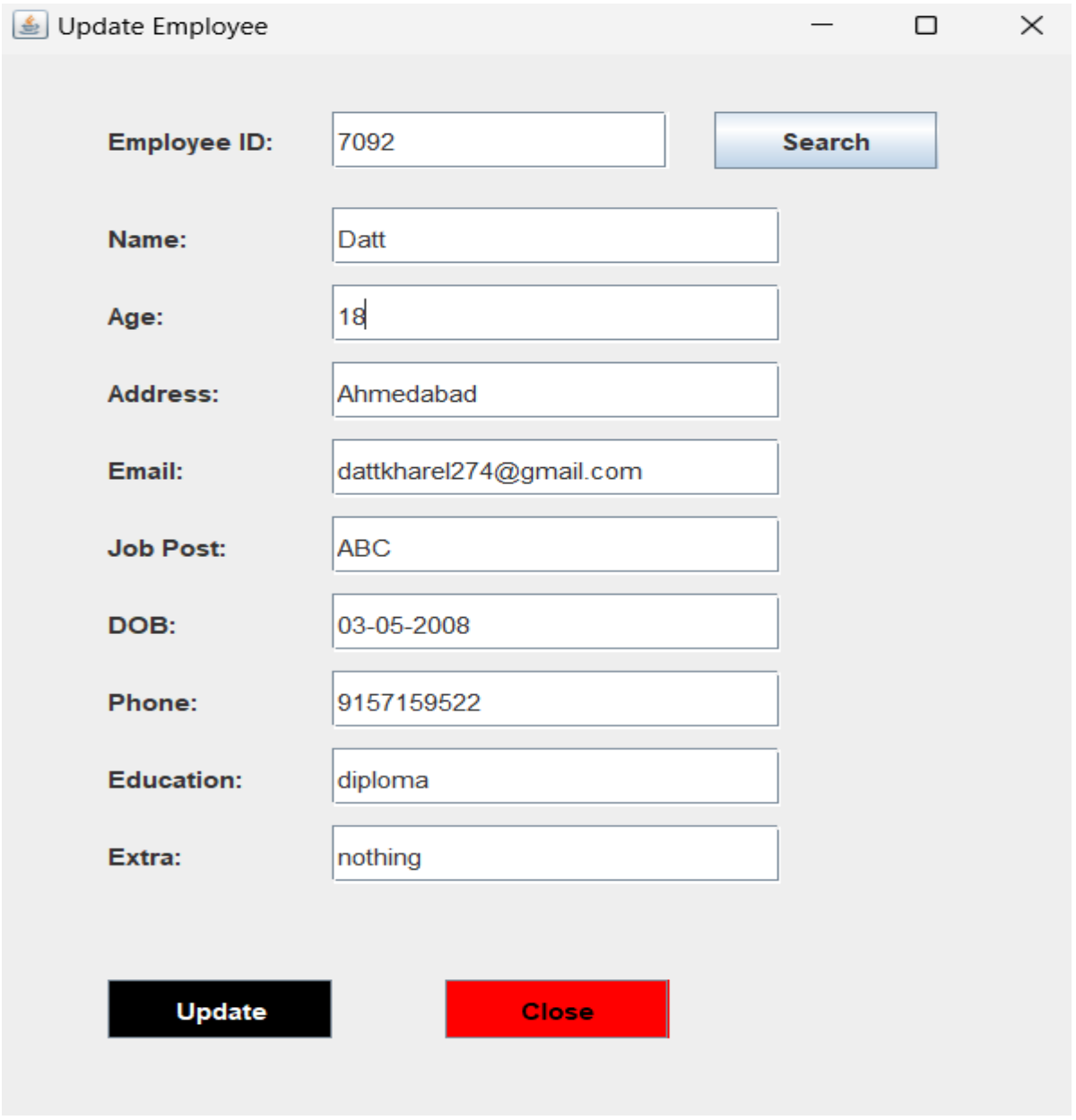

Employee Homepage
Profile
Leave
Manage
Salary
Attendance
Delete
Exit

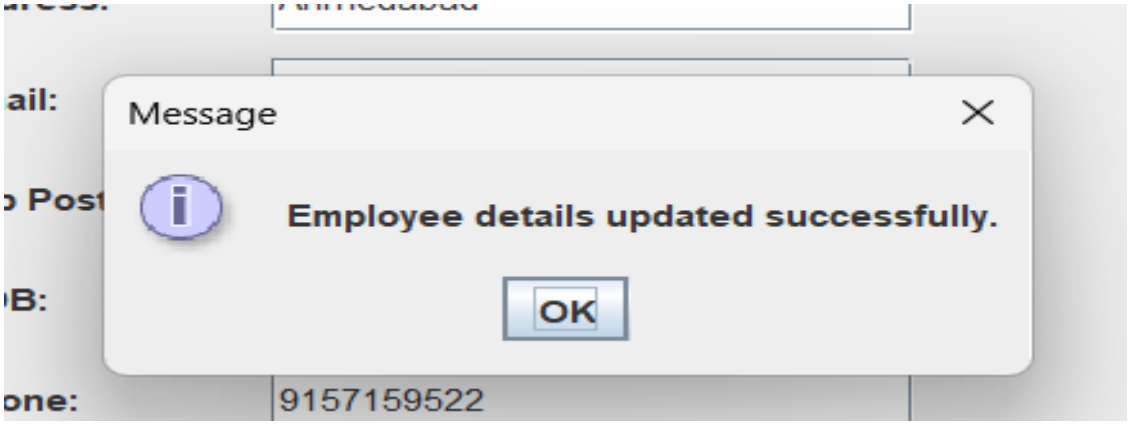

Profile Options
View Profile
Add Profile


Employee Detail
Enter ID:
7092
Search
Name: Datt
Age: 16
Address: Ahmedabad
Email: dattkharel274@gmail.com
Job Post: ABC
DOB: 03-05-2008
Phone: 9157159522
Education: diploma
Extra: nothing
Close

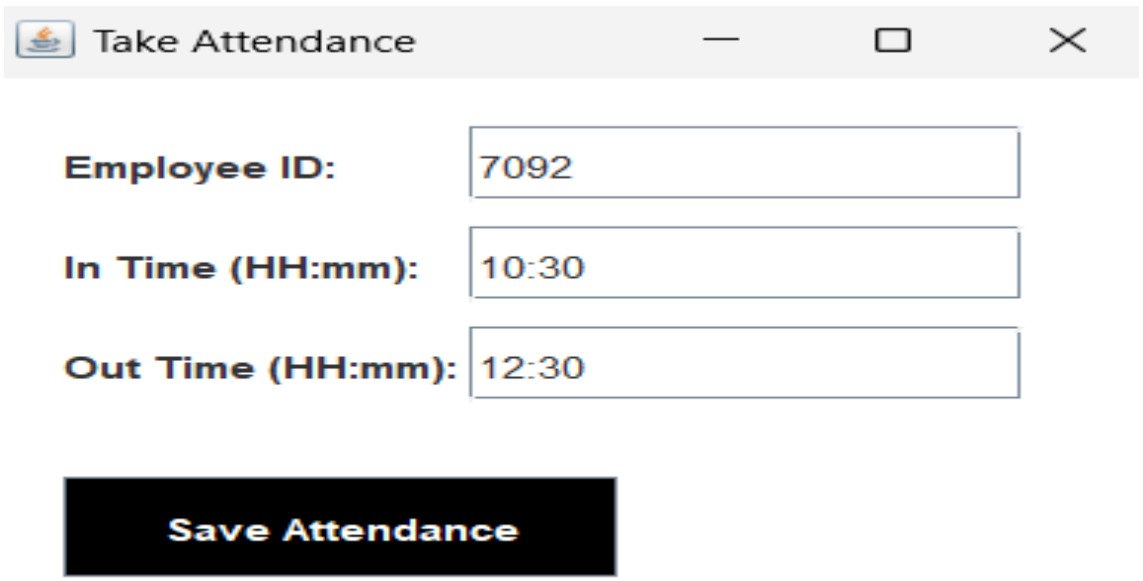

New Employee Details
Employee ID:
7090
Job Post:
Student
Email:
shubh@gmail.com
Address:
Ahmedabad
Age:
17
Name:
Shubh
DOB:
03-04-2008
Phone:
1234567890
Education:
student
Extra:
nothing
Submit
Cancel

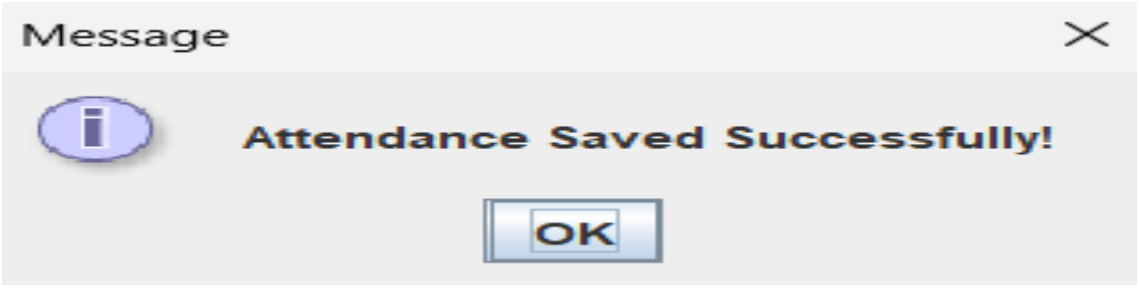

Address:
Ahmedabad
Message
Employee Added Successfully!
OK
Phone:
1234567890


Update Employee
Employee ID:
7092
Search
Name:
Datt
Age:
18
Address:
Ahmedabad
Email:
dattkharel274@gmail.com
Job Post:
ABC
DOB:
03-05-2008
Phone:
9157159522
Education:
diploma
Extra:
nothing
Update
Close


Message
Employee details updated successfully.
OK
9157159522


Attendance Options
Take Attendance
Get Attendance List


Take Attendance
Employee ID:
7092
In Time (HH:mm):
10:30
Out Time (HH:mm):
12:30
Save Attendance



Attendance List

Enter Employee ID: 7092

Get Attendance | Show All

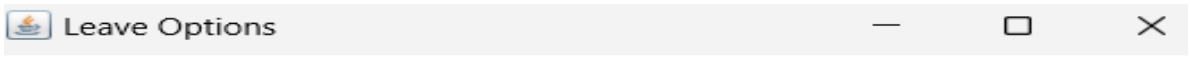
Emp ID	In Time	Out Time	Total Hours	Salary	Date
7092	10:30	11:30	1.0	₹500.0	2025-04-12
7092	12:00	13:30	1.5	₹750.0	2025-04-12
7092	10:00	10:10	0.16666666666666666	₹83.33333333333333	2025-04-12
7092	10:00	10:30	0.5	₹250.0	2025-04-12
7092	10:30	12:30	2.0	₹1000.0	2025-04-13

Attendance List

Enter Employee ID:

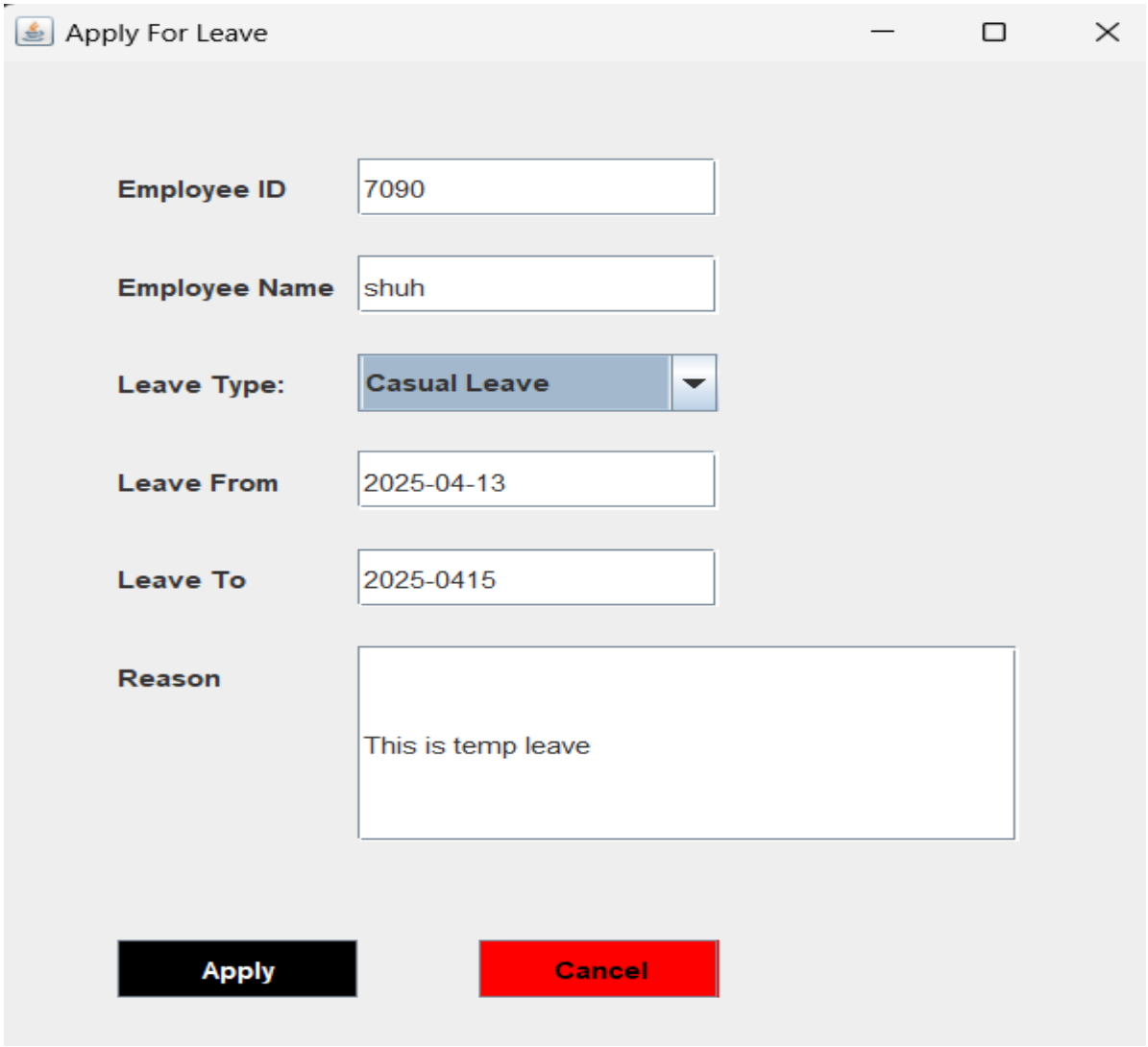
Get Attendance | Show All

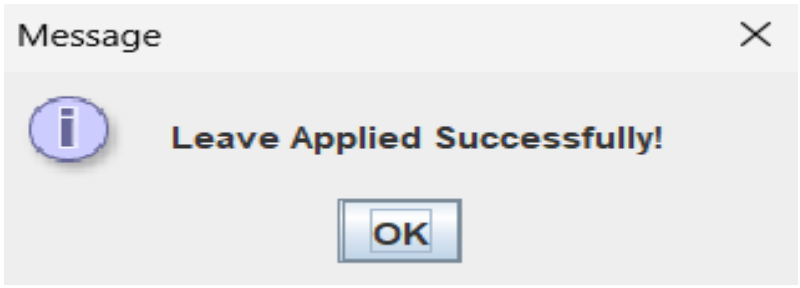
Emp ID	In Time	Out Time	Total Hours	Salary	Date
7092	10:30	11:30	1.0	₹500.0	2025-04-12
7092	12:00	13:30	1.5	₹750.0	2025-04-12
7092	10:00	10:10	0.16666666666666666	₹83.33333333333333	2025-04-12
7092	10:00	10:30	0.5	₹250.0	2025-04-12
7092	10:30	12:30	2.0	₹1000.0	2025-04-13
7090	10:00	11:30	1.5	₹750.0	2025-04-13

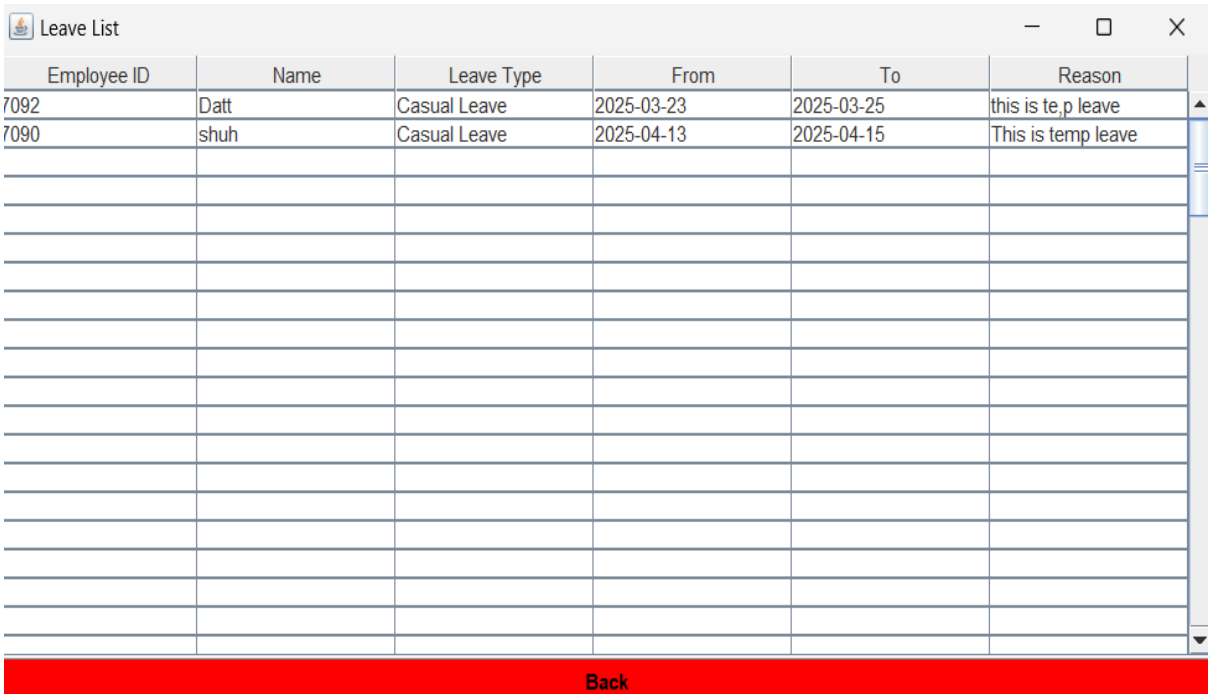

Leave Options

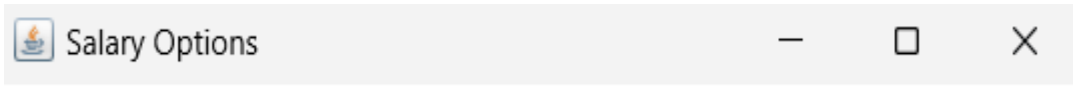

Apply For Leave
Get Leave List


Leave Type:
Casual Leave
Casual Leave
Sick Leave
Maternity Leave
Paternity Leave
Compensatory Leave
Leave From
Leave To

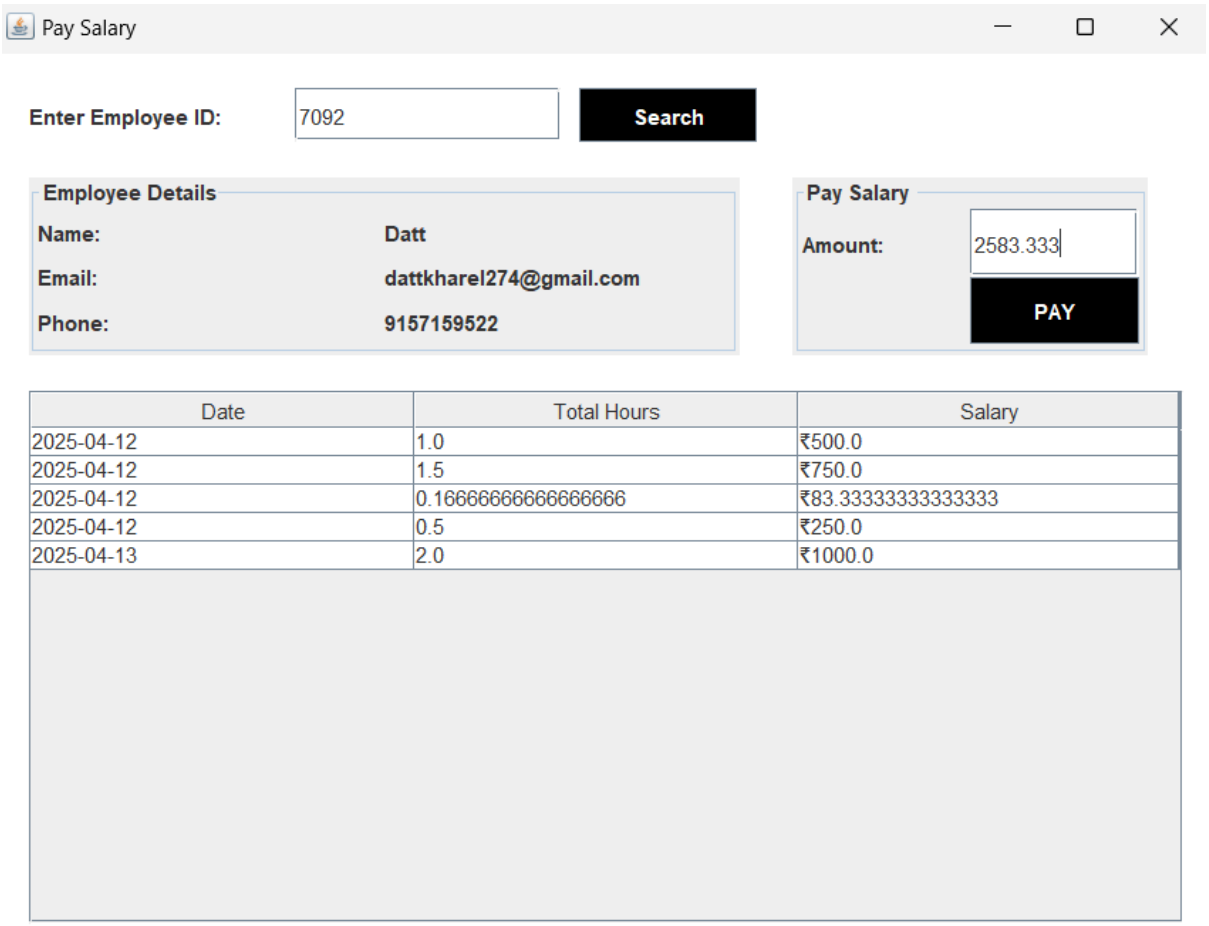

Apply For Leave
Employee ID
7090
Employee Name
shuh
Leave Type:
Casual Leave
Leave From
2025-04-13
Leave To
2025-0415
Reason
This is temp leave
Apply
Cancel

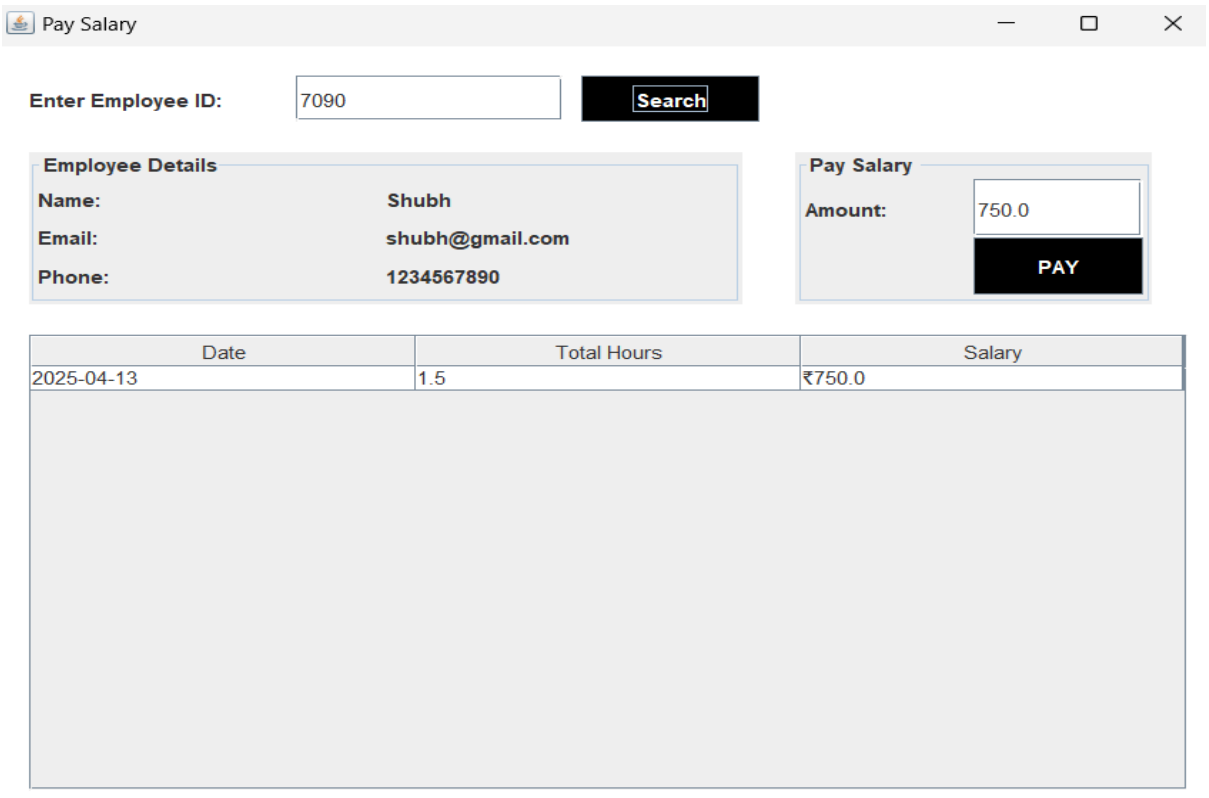

Message
Leave Applied Successfully!
OK

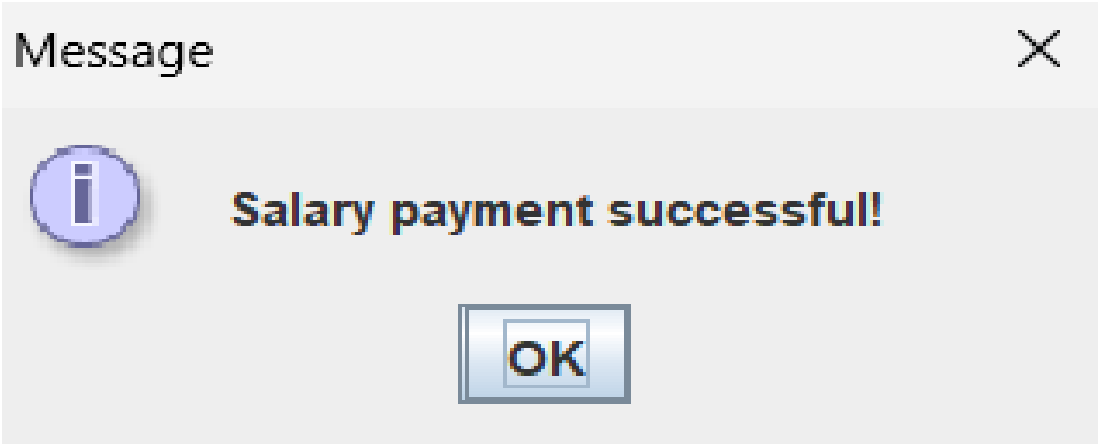

Leave List
Employee ID
Name
Leave Type
From
To
Reason
7092
Datt
Casual Leave
2025-03-23
2025-03-25
this is te,p leave
7090
shuh
Casual Leave
2025-04-13
2025-04-15
This is temp leave
Back


Salary Options


Pay Salary
Salary History

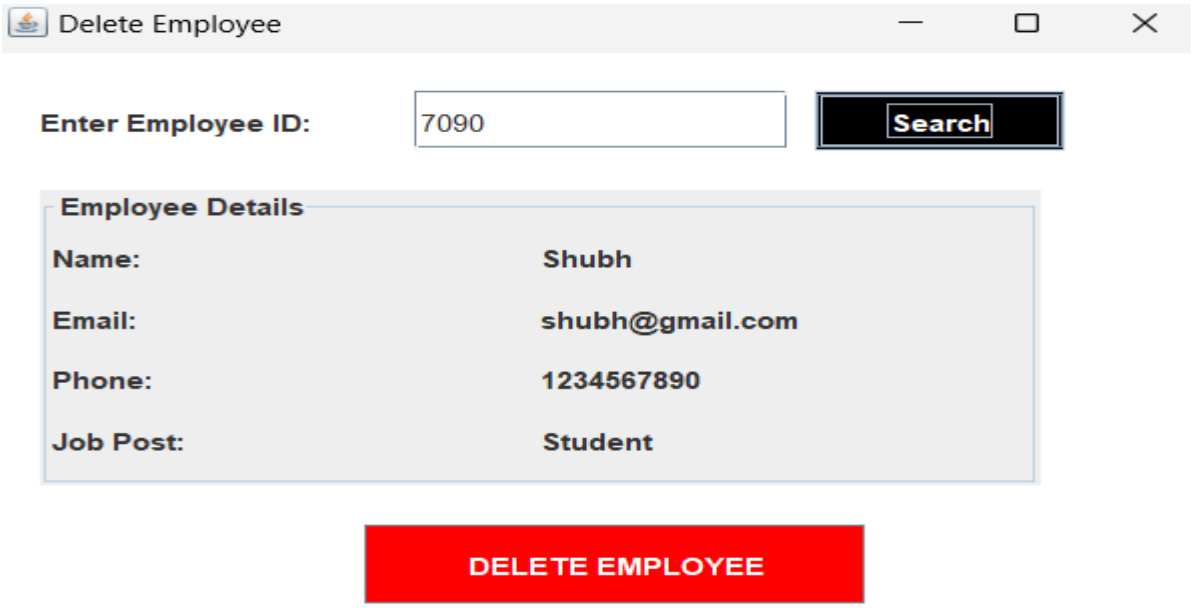

Pay Salary
Enter Employee ID:
7092
Search
Employee Details
Name:
Datt
Email:
dattkharel274@gmail.com
Phone:
9157159522
Pay Salary
Amount:
2583.333
PAY
Date
Total Hours
Salary
2025-04-12
1.0
₹500.0
2025-04-12
1.5
₹750.0
2025-04-12
0.16666666666666666
₹83.33333333333333
2025-04-12
0.5
₹250.0
2025-04-13
2.0
₹1000.0

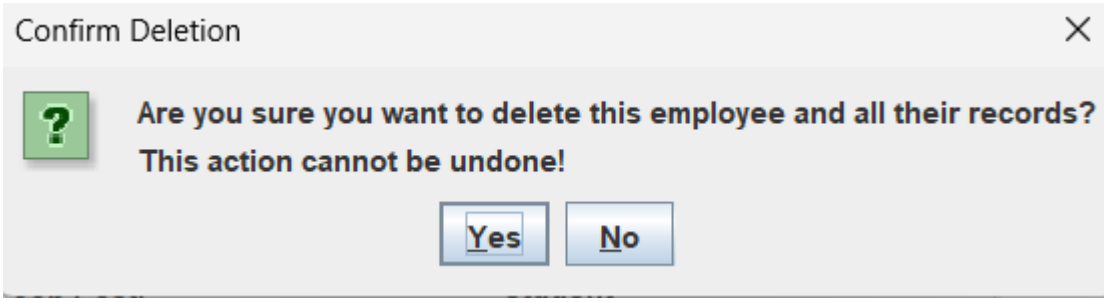

Pay Salary
Enter Employee ID:
7090
Search
Employee Details
Name:
Shubh
Email:
shubh@gmail.com
Phone:
1234567890
Pay Salary
Amount:
750.0
PAY
Date
Total Hours
Salary
2025-04-13
1.5
₹750.0

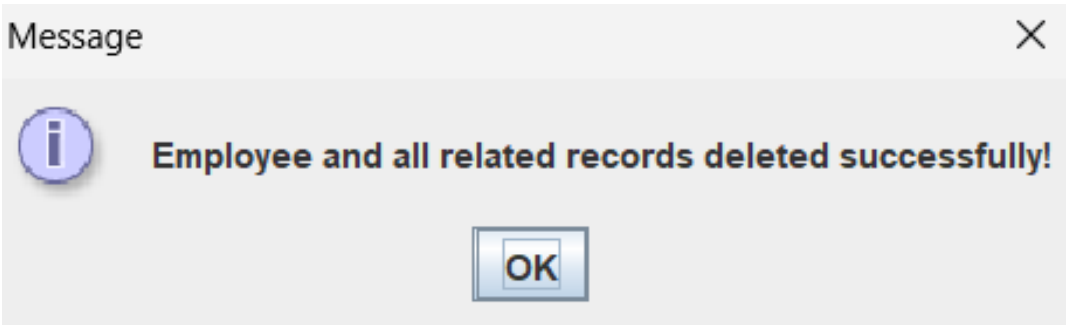


Salary Payment History

Payment ID	Employee ID	Amount	Payment Date
1	7092	₹1583.333	2025-04-12
2	7092	₹1583.333333	2025-04-13
3	7090	₹750.0	2025-04-13







Purpose of EXIT Button:

○ Closes the Application:

- **When clicked, it exits the entire system gracefully.**
- **It terminates the currently running Java Swing application window (JFrame) and shuts down the app.**

9. Limitation & Future Scope

9.1 Limitations of the Current System

Despite the comprehensive functionalities provided in the Employee Management System, there are certain limitations in the current version:

- **Limited User Roles**: Currently, the system primarily supports a single admin role. There is no provision for different user roles such as HR, Manager, or Employee with varied access privileges.
- **Lack of Web Interface**: The application is developed using Java Swing, which limits accessibility to desktop environments only. Users cannot access the system via a browser or mobile device.
- **No Notification System**: There is no automated notification system (like email or SMS alerts) for actions such as leave approvals, salary payments, or attendance irregularities.
- **Manual Attendance Marking**: Attendance is manually entered by the admin or manager, which can lead to errors. Integration with biometric or RFID systems is not present.
- **Limited Analytics**: The system lacks advanced reporting or analytics features like graphs, trends, or data visualization for managerial insights.
- **No Backup or Recovery Mechanism**: Currently, there is no built-in functionality for automatic database backups or disaster recovery.
- **UI Design Limitations**: Although functional, the UI is basic and may not be intuitive for users without technical knowledge.

9.2 Future Enhancements

To overcome the limitations and make the system more efficient, several future enhancements can be considered:

- **Role-Based Access Control (RBAC)**: Implementing different access levels for admin, HR, employees, and managers to ensure secure and customized experiences.
- **Web-Based and Mobile App Versions**: Migrating to a full-stack web application or creating a hybrid app to support wider accessibility on any device.
- **Automated Notifications**: Integrating an email/SMS notification system to inform users of events like approved leaves, salary deposits, and holidays.
- **Biometric Integration**: Implementing biometric or card-based attendance to reduce manual errors and ensure accurate tracking of employee presence.
- **Advanced Reporting**: Adding dashboards and data visualization tools for real-time insights on employee performance, attendance, and salary trends.
- **Backup and Restore Feature**: Incorporating a scheduled backup and data recovery system to protect against data loss.
- **Modern UI Design**: Revamping the interface using modern frameworks like JavaFX or transitioning to a web front-end like Angular or React for better UX.

10. Conclusion & Discussion

The **Employee Management System** project is a robust and modular desktop-based solution developed using **Java Swing** for the front end and **MySQL** as the back-end database. It automates multiple HR tasks such as employee record management, attendance tracking, leave management, and salary calculation. The system is particularly beneficial for small to medium-sized businesses looking to digitize their HR operations without complex infrastructure.

The project has successfully achieved the following goals:

- Provided a central platform to manage all employee-related data.
- Simplified salary calculations using automated working hour calculations excluding break times.
- Offered modules for inserting, updating, deleting, and viewing employee records.
- Designed a clean, user-friendly interface using Java Swing components.

Through the process of **requirement gathering, system analysis, design, implementation, and testing**, the system has been carefully planned and executed. Extensive **unit and integration testing** was carried out to ensure stability and functionality of each module.

However, the discussion also revealed that there is room for improvement in terms of scalability, user accessibility, and integration with external tools. These future scopes provide direction for the next versions of the system.

In conclusion, this project serves as a foundational prototype for an efficient Employee Management System. With further enhancements and optimizations, it has the potential to evolve into a powerful enterprise-level solution for human resource management.