****

**Project Report on**

**Payroll Payroll Program**

**Submitted to**

**VISHWAKARMA UNIVERSITY, PUNE**

**Under the Initiative of**

**Contemporary Curriculum, Pedagogy, and Practice**

**(C2P2)**

**PROJECT BASED LEARNING - PYTHON**

**By**

|  |  |
| --- | --- |
| **Name of the Student** | **: Gaurav A. Guddeti** |
| **SRN. No.** | **: 31230420** |
| **Class/Div.** | **: SY / C** |
| **Roll no.** | **: 05** |

**Faculty In-charge: Prof. Jayendra Jadhav**

**Department of Artificial Intelligence**

**Faculty of Science & Technology**

**Academic Year**

**2024-2025 Term-II**

1. ProblemStatement
   1. Problem Statement Sr. No:
   2. Project Title:
   3. Problem Statement:
2. Introduction
3. Methodology
4. Implementation and Results
5. Outcomeof the Project
6. Conclusion
7. References

1. **Problem Statement**

1.1. **Problem Statement Sr. No:** 5

1.2. **Project Title:** Employee Payroll Generator

1.3. **Problem Statement:**

A small business requires an application to calculate employee salaries, including overtime pay and tax deductions. The application needs to store employee details, hourly rates, and hours worked using objects. It should use control structures to calculate gross pay, tax deductions, and net salary, with a report generation feature.

1. **Introduction**

The Employee Payroll Generator is a GUI-based Java application designed to streamline the payroll management process for small businesses. This application provides a user-friendly interface for calculating employee salaries, handling overtime payments, and managing tax deductions. The system is built using Java Swing for the graphical interface and implements object-oriented programming principles for efficient data management.

1. **Methodology**

The project follows a structured development approach:

1. Analysis Phase:

* Requirements gathering
* System design
* Data structure planning

1. Design Phase:

* Class design (Employee, PayrollCalculator, PayrollGUI)
* Interface design
* Calculation logic design

1. Implementation Phase:

* GUI development using Java Swing
* Core functionality implementation
* Testing and validation

1. **Implementation and Results**

The implementation includes:

1. Core Classes:
2. Employee.java:

public class Employee {

private String employeeId;

private String name;

private double hourlyRate;

private double hoursWorked;

private double overtimeHours;

private double grossPay;

private double taxDeductions;

private double netSalary;

// Constructor

public Employee(String employeeId, String name, double hourlyRate) {

this.employeeId = employeeId;

this.name = name;

this.hourlyRate = hourlyRate;

}

// Getters and Setters

public String getEmployeeId() { return employeeId; }

public void setEmployeeId(String employeeId) { this.employeeId = employeeId; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public double getHourlyRate() { return hourlyRate; }

public void setHourlyRate(double hourlyRate) { this.hourlyRate = hourlyRate; }

public double getHoursWorked() { return hoursWorked; }

public void setHoursWorked(double hoursWorked) {

this.hoursWorked = hoursWorked;

}

public double getOvertimeHours() { return overtimeHours; }

public void setOvertimeHours(double overtimeHours) {

this.overtimeHours = overtimeHours;

}

public double getGrossPay() { return grossPay; }

public void setGrossPay(double grossPay) { this.grossPay = grossPay; }

public double getTaxDeductions() { return taxDeductions; }

public void setTaxDeductions(double taxDeductions) { this.taxDeductions = taxDeductions; }

public double getNetSalary() { return netSalary; }

public void setNetSalary(double netSalary) { this.netSalary = netSalary; }

}

1. PayrollCalculator.java:

public class PayrollCalculator {

private static final double OVERTIME\_RATE = 2.0; // Double pay for overtime

private static final double TAX\_RATE = 0.1; // 10% tax rate

public static void calculatePayroll(Employee employee) {

double regularPay = employee.getHoursWorked() \* employee.getHourlyRate();

double overtimePay = employee.getOvertimeHours() \* employee.getHourlyRate() \* OVERTIME\_RATE;

double grossPay = regularPay + overtimePay;

double taxDeductions = grossPay \* TAX\_RATE;

double netSalary = grossPay - taxDeductions;

employee.setGrossPay(grossPay);

employee.setTaxDeductions(taxDeductions);

employee.setNetSalary(netSalary);

}

}

1. PayrollGUI.java

public class PayrollGUI extends JFrame {

private ArrayList<Employee> employees = new ArrayList<>();

private JTextField idField, nameField, rateField, hoursField, overtimeField;

private JTextArea outputArea;

private DecimalFormat df = new DecimalFormat("#,##0.00");

// Constructor

public PayrollGUI() {

setTitle("Employee Payroll Generator - Indian Rupee Edition");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout(10, 10));

// Main Panel with padding

JPanel mainPanel = new JPanel(new BorderLayout(10, 10));

mainPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

mainPanel.setBackground(Color.WHITE);

// Add components

add(createInputPanel(), BorderLayout.NORTH);

add(createButtonPanel(), BorderLayout.CENTER);

add(createOutputPanel(), BorderLayout.SOUTH);

pack();

setLocationRelativeTo(null);

setMinimumSize(new Dimension(600, 700));

}

// Calculate Payroll Method

private void calculatePayroll() {

try {

String id = idField.getText().trim();

String name = nameField.getText().trim();

double rate = Double.parseDouble(rateField.getText().trim());

double hours = Double.parseDouble(hoursField.getText().trim());

double overtime = Double.parseDouble(overtimeField.getText().trim());

if (id.isEmpty() || name.isEmpty()) {

showError("Please fill all fields");

return;

}

if (rate <= 0 || hours < 0 || overtime < 0) {

showError("Please enter valid positive numbers");

return;

}

Employee employee = new Employee(id, name, rate);

employee.setHoursWorked(hours);

employee.setOvertimeHours(overtime);

PayrollCalculator.calculatePayroll(employee);

employees.add(employee);

displayEmployeeDetails(employee);

clearInputFields();

} catch (NumberFormatException ex) {

showError("Please enter valid numbers for rate and hours");

}

}

}

1. **Key Features Implementation:**
2. Input Validation:

private void validateInput() {

if (id.isEmpty() || name.isEmpty()) {

showError("Please fill all fields");

return;

}

if (rate <= 0 || hours < 0 || overtime < 0) {

showError("Please enter valid positive numbers");

return;

}

}

b) Salary Calculation:

public static void calculatePayroll(Employee employee) {

double regularPay = employee.getHoursWorked() \* employee.getHourlyRate();

double overtimePay = employee.getOvertimeHours() \* employee.getHourlyRate() \* OVERTIME\_RATE;

double grossPay = regularPay + overtimePay;

double taxDeductions = grossPay \* TAX\_RATE;

double netSalary = grossPay - taxDeductions;

employee.setGrossPay(grossPay);

employee.setTaxDeductions(taxDeductions);

employee.setNetSalary(netSalary);

}

**Results:**

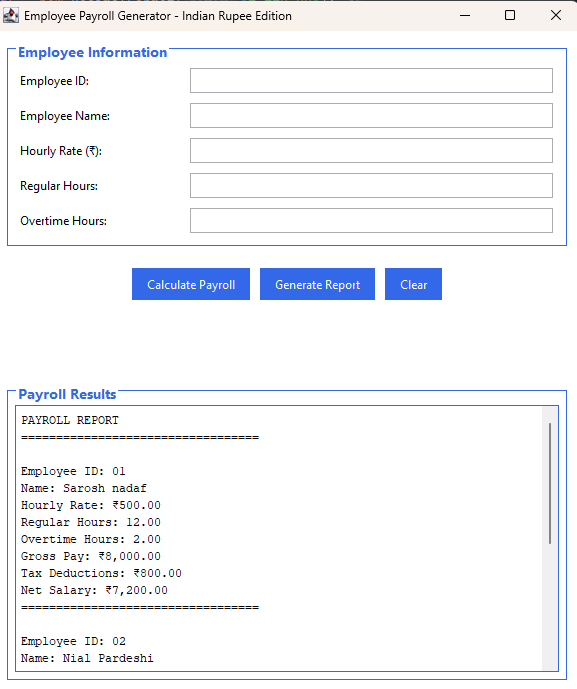
The implementation successfully provides:

* A user-friendly interface for data entry
* Accurate salary calculations including overtime
* Proper tax deduction processing
* Comprehensive report generation
* Input validation and error handling
* Professional formatting of currency values
* Efficient data management using ArrayList

The code demonstrates proper implementation of:

* Object-oriented principles
* GUI development using Java Swing
* Exception handling
* Data validation
* Report formatting
* Currency handling

**GUI OUTCOME:**

****

1. **Outcome of the project**

The project successfully achieved:

1. Functional Outcomes:

* Automated salary calculation
* Overtime pay management
* Tax deduction processing
* Report generation

1. Technical Outcomes:

* Implementation of GUI using Java Swing
* Object-oriented design
* Data validation and error handling
* Professional-grade formatting

1. Business Outcomes:

* Streamlined payroll process
* Reduced calculation errors
* Improved efficiency
* Better record management

1. **Conclusion**

The Employee Payroll Generator project successfully addresses the payroll management needs of small businesses. The application demonstrates the practical application of Java programming concepts and provides a user-friendly solution for salary calculations. The implementation of object-oriented principles, GUI development, and business logic showcases the integration of technical skills with real-world business requirements.

1. **References**

1. Java Swing Documentation

1. Object-Oriented Programming Principles
2. GUI Design Best Practices
3. Payroll Management Guidelines
4. Java Programming Standards
5. Software Development Life Cycle Documentation