I. INTEGRATED SITUATION

Project Description

The Government of Rwanda employs thousands of employees across its different institutions. The Government has provided Enterprise Resource Planning (ERP) systems to assist each institution in managing its finance and human resources. Among the core modules of ERP include, but are not limited to:

a) **Employee Management** — which enables centralized storage of employee details, including personal information and professional/employment details.
b) **Payroll Management System** — a core module of ERP tailored for managing employee compensation by calculating salaries based on institutional rules and regulations, indicating taxes, deductions, and base salary.

The existing payroll management system uses outdated rates of taxes and deductions, which include:

- Employee tax: 30%
- Pension: 3%
- Medical insurance: 5%
- Housing: 14%
- Transport: 14%
- Others: 5%

All of these are deducted from the base salary.

Starting from **January 2025**, the Government, through the Rwanda Social Security Board, introduced a **new pension rate** changing it from 3% to **6%**. With this update, the Government wants to upgrade the ERP system starting with the updated deduction rates to compute new salaries.

TASKS

As a Java developer, you have been hired to develop the **backend** for the Enterprise Resource Planning system, starting with **Payroll Management** and **Employee Management**, with the following minimum features:

Task 1: Design Payroll Management System Database using Spring Data JPA

Design the following tables:

- **Employee**:
  Properties: `code`, `firstName`, `lastName`, `email`, `password`, `roles`, `mobile`, `dateOfBirth`, `status` (active or disabled)

- **Employment**:
  Properties: `code`, `employeeId`, `department`, `position`, `baseSalary`, `status` (active or inactive), `joiningDate`

- **Deductions**:
  Properties: `code`, `deductionName`, `percentage`

- **Payslip**:
  Properties: `id`, `employee`, `houseAmount`, `transportAmount`, `employeeTaxedAmount`, `pensionAmount`, `medicalInsuranceAmount`, `otherTaxedAmount`, `grossSalary`, `netSalary`, `month`, `year`, `status` (pending or paid)

Task 2: Employee and Employment Management

- Provide endpoints (CRUD APIs) for managing employee personal information.
- Each employee registered is a user of the system.
- Employees are managed with role-based security.
- Implement **Authentication and Authorization using JWT**.

Login details:

- Employees authenticate using `email` (as username) and `password`.

Roles assigned to Employees:

- `ROLE_MANAGER`: Can process salary and add employee details.
- `ROLE_ADMIN`: Can approve salary.
- `ROLE_EMPLOYEE`: Can view his/her details, download payslip, and view pending salary payments.

Task 3: Deductions and Taxes Management

Generate endpoints for managing deduction percentages. The following sample deductions should be applied:

No	Deduction Name	Percentage
1	Employee Tax	30%
2	Pension	6%
3	Medical Insurance	5%
4	Others	5%
5	Housing	14%
6	Transport	14%

Task 4: Payroll Generation and Payslip

The manager can start the payroll process for a given month and year based on base salary and deductions. The system should compute and generate the salary of all **active** employees as payroll. Ensure that all deductions do not exceed the gross salary.

Each individual employee can view their payslip for a given month and year. The manager can also view all payslips for a given month/year.

Note:

- The **base salary** is used to compute all deductions, gross salary, and net salary.

Example Calculation:

Given the base salary of employee *David* as 70,000 RWF:

- Gross Salary = baseSalary + housing + transport
  = 70,000 + (70,000 * 14%) + (70,000 * 14%)
  = 70,000 + 9,800 + 9,800 = **89,600**

- Net Salary = grossSalary - (employeeTax + pension + medicalInsurance + others)
  = 89,600 - (21,000 + 4,200 + 3,500 + 3,500) = **57,400**

Sample Payslip - December 2024 (RCA Employees)

Emp ID	Name	Base	Housing	Transport	Gross	Tax	Pension	Medical	Others	Net	Status	Month	Year
123	Mugabo	70000	9800	9800	89600	21000	4200	3500	3500	57400	Pending	12	2025
234	Micheline	35000	4900	4900	44800	10500	2100	1750	1750	28700	Pending	12	2025

Task 5: Messaging Feature using Database Routines

a) When the ADMIN approves the payroll using the appropriate endpoint (updating the payslip status from `pending` to `paid`), configure a **database trigger** to generate a message to be sent to each employee. Add a `message` table that includes:

- `employee`
- `message`
- `month-year` of message sent

b) The message should be **sent to the employee's email**.

Format of the message:

Dear <FIRSTNAME>, your salary for <MONTH>/<YEAR> from <INSTITUTION> amounting to <AMOUNT> has been credited to your account <EMPLOYEE ID> successfully.

Additional Notes:

- Include but are not limited to:
    - Database ERD schema
    - Application architecture
    - List of POJOs
    - Justification of chosen technologies
- The backend **must be designed using Spring Boot**, and **Spring Data JPA** should be used for database configuration and API generation.
- Design a **Spring Boot Flow Diagram** showing system functionality.
- Records of users, employee details, deductions, and transactions can be added manually via Postman, Swagger, main class, or at the DBMS level.
- Document your APIs using **Swagger UI**.
- Use **JWT** for authentication and authorization.
- Prevent duplicate payroll generation for the same employee in the same month/year.

Time Allocation:

- In the first hour: Read the problem carefully, design the solution, and write down appropriate plans.
- **Total time allocated: 5 hours**