A logo of a university

AI-generated content may be incorrect.A logo of a university

AI-generated content may be incorrect.

Holy Angel University

School of Computing

Computer Science Department

**DESIGN AND IMPLEMENTATION OF**

**PROGRAMMING LANGUAGES &**

**INFORMATION MANAGEMENT FINAL PROJECT:**

***Haskell Payroll System***

Final Output/Project

2nd Semester, School Year 2024-2025

**Submitted by:**

Del Mundo, Guiane Carlo

Lumba, Nelwyn Jairoh

Pugal, Reiven Cuert

Rivera, Kurt Francis

CS-202

DESIGN AND IMPLEMENTATION OF PROGRAMMING LANGUAGES &

INFORMATION MANAGEMENT

**Submitted to:**

Mrs. Ma. Louella D. Salenga

MARCH 2025

**Introduction**

A payroll is a list of employees including the wages or salaries due to each employee. Computer payroll system as personnel system also provide broad range of personnel and payroll processing like keeping employees records, pay band, savings deduction, taxes and others using a computer database (Elekwa, Eme, 2013).

As stated by Kaur, et al, a payroll is a critical operation for every organization to pay employee accurately their salary and enrollments on time. The idea of taking control of employees pay calculations are quite tedious if done manually and require more effort and time mainly for big organizations. Hence if this process is automated, it would be of great benefit as it would require less time to calculate the salary of the employees.

Haskell is a purely functional language that allows programmers to rapidly develop software that is clear, concise and correct (Hutton, 2016). Every function in Haskell is a function in the mathematical sense (i.e., "pure"). Even side-effecting IO operations are but a description of what to do, produced by pure code. There are no statements or instructions, only expressions which cannot mutate variables (local or global) nor access state like time or random numbers (Haskell.org, n.d.). With this in mind, the researchers have utilized Haskell in a way that it can transform and save data, complete with conditional and iterative statements.

This project is a simple console-based program written in Haskell, a functional programming language that simulates a payroll system. The application that the researchers have developed support adding, editing and deleting employee records. The application can also automatically calculate their salaries based on their hourly rate, hours worked, and will deduct 10% on their salary should the administrator wish to do so. It ensures data integrity as the program uses file handling to save and retrieve employee data.

**Scope**

* Add, edit, and delete employees
* Compute salaries with a 10% deduction
* Show all employee records
* Persistent storage: Saves employee data in a file (payroll.db).
* Garbage collection: Ensures data is saved before exit.

**Limitations**

* Lack of Constraints: The program utilizes a simple text file instead of a relational database, making it have zero constraints, making it easy to break and exploit.
* Lack of Graphical Interface: With Haskell having little to no documentation about frontend development using the language, the researchers have decided to strictly use the console in typing commands to make the program useable.
* Lack of Multi-User Access: Since it is only a console-based program, there is no way for multiple users to interact with the system simultaneously.

**How it Works**

Upon running the program, the system loads employee records from payroll.db. Users interact with the system via a text-based menu, selecting options to manage payroll data. Changes are automatically saved.

**Usage**

Run the Haskell program, and follow the menu options:

1. **Add Employee** - Enter employee details.
2. **Edit Employee** - Modify employee information.
3. **Delete Employee** - Remove an employee.
4. **Compute Salary** - View an employee's salary breakdown.
5. **Show All Employees** - List all stored employees.
6. **Exit** - Saves data and exits the program.

**Data Types**

* Employee - Represents an employee with fields: empId, empName, empPosition, empRate, and empHours.
* PayrollDB - An abstract data type (ADT) that encapsulates employee management.

**Main Functions**

* main - Initializes the program and loads data.
* mainMenu - Displays menu options.
* loadPayroll - Reads employee records from payroll.db.
* savePayroll - Saves employee records to a file.
* addEmployee - Adds a new employee.
* editEmployee - Modifies employee details.
* deleteEmployee - Removes an employee.
* computeSalary - Calculates an employee’s salary with deductions.
* showEmployees - Lists all employees.

**Business Rules for Database**

The business rules for this payroll system are:

1. Each employee has a unique ID.
2. Employees have an hourly rate and worked hours, which determine their salary.
3. Employees can be added, edited, or deleted by authorized users.
4. Salary is calculated as:

Basic Salary = Hourly Rate x Hours Worked

Deductions = Basic Salary x 10 % Deductions = Basic Salary x 10%

Net Salary = Basic Salary − Deductions Net Salary = Basic Salary − Deductions

**Normalization of Database**

To ensure efficient data organization, we apply 3rd Normal Form (3NF):

**1st Normal Form (1NF)**

* Eliminate repeating groups.
* Each field contains atomic values.

**2nd Normal Form (2NF)**

* Remove partial dependencies.
* Ensure each non-key attribute depends on the whole primary key.

**3rd Normal Form (3NF)**

* Remove transitive dependencies.
* Ensure non-key attributes depend only on the primary key.

Final Table Structure (3NF):

1. Employees Table → Stores employee information.
2. Payroll Table → Stores salary details.
3. Users Table → Manages system access.

**Entity Relationship Diagram**

**A screenshot of a diagram

AI-generated content may be incorrect.**

are in

**Data Dictionary of Tables**

**Employees Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description |
| empId | INT | - | PRIMARY KEY | Unique Employee ID |
| empName | VARCHAR | 100 | NOT NULL | Employee Full Name |
| empPosition | VARCHAR | 50 | NOT NULL | Job Position |
| empRate | FLOAT | - | NOT NULL | Hourly Rate |
| empHours | FLOAT | - | NOT NULL | Hours Worked |

**Payroll Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description |
| payrollId | INT | - | PRIMARY KEY | Unique Payroll ID |
| empId | INT | - | FOREIGN KEY | Employee ID Reference |
| basicSalary | FLOAT | - | NOT NULL | Calculated Basic Salary |
| deductions | FLOAT | - | NOT NULL | 10% Deductions |
| netSalary | FLOAT | - | NOT NULL | Final Salary |

**Users Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description |
| userId | INT | - | PRIMARY KEY | Unique User ID |
| username | VARCHAR | 50 | UNIQUE, NOT NULL | System Login Name |
| password | VARCHAR | 255 | NOT NULL | Hashed Password |

**Queries, Subquries, Joins**

**Create Tables**

A screenshot of a computer program

AI-generated content may be incorrect.

**Insert Employee Data**

****

**Compute Salary**

**A white background with black text

AI-generated content may be incorrect.**

**Inner Join: Employee & Payroll**

**A close up of a computer screen

AI-generated content may be incorrect.**

**Code Snippets**

**A computer screen shot of a code

AI-generated content may be incorrect.**

**Database Structure**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Conclusion**

Despite Haskell being a “purely mathematical” functional language, it is still capable of utilizing iterative and conditional statements to make a simple menu-based program that can handle file input and output, memory management, ADT and encapsulation in an imperative-style programming paradigm, even though it is classified as declarative programming.

**References:**

Elekwa, N. N., & Eme, O. I. (2013). An analysis of computerized accounting and payrolling system on monthly emolument in Nigerian local government. *International Journal of Accounting Research*, *1*(3), 16-28.

Hutton, G. (2016). Programming in haskell. *Cambridge University Press.*

Kaur, P., & Grover, D. (2012). Computer based payroll system implementation for E-governance at Punjab agricultural university. *International Journal of Engineering Research and Development*, 5(3), 55-60.

Haskell.org (n.d.). Features of Haskell