Holy Angel University

DESIGN AND IMPLEMENTATION OF PROGRAMMING LANGUAGES FINAL PROJECT:

Haskell Payroll System

Final Output/Project

2nd Semester, S.Y 2024-2025

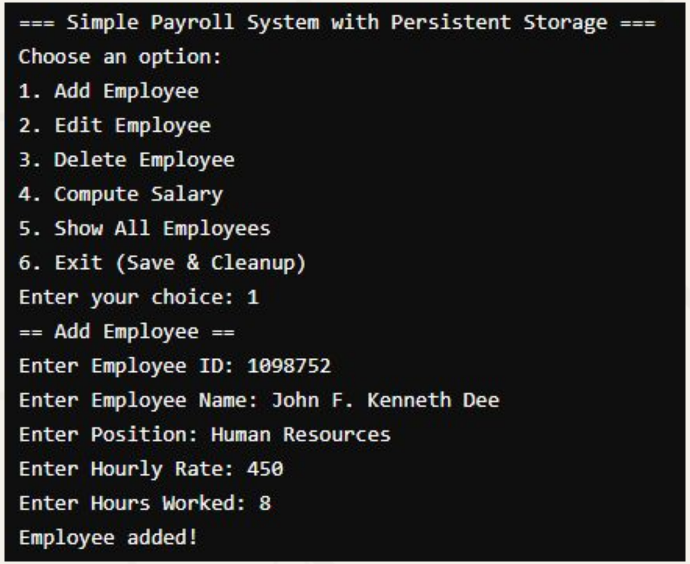
Del Mundo, Guiane Carlo
Lumba, Nelwyn Jairoh
Pugal, Reiven Cuert
Rivera, Kurt Francis

CS - 202

INTRODUCTION

Overview

The Haskell program implements a simple payroll system with persistent storage. It allows users to manage employee records, compute salaries, and maintain data across sessions using file storage.

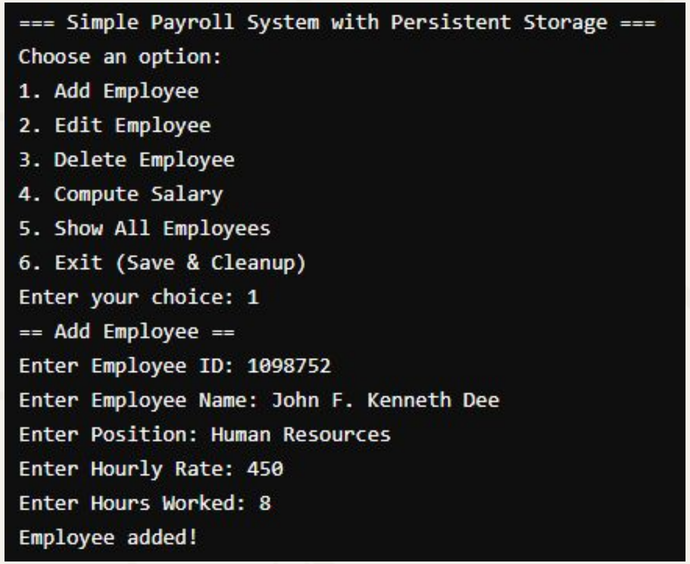


```
=== Simple Payroll System with Persistent Storage ===
Choose an option:
1. Add Employee
2. Edit Employee
3. Delete Employee
4. Compute Salary
5. Show All Employees
6. Exit (Save & Cleanup)
Enter your choice: 1
== Add Employee ==
Enter Employee ID: 1098752
Enter Employee Name: John F. Kenneth Dee
Enter Position: Human Resources
Enter Hourly Rate: 450
Enter Hours Worked: 8
Employee added!
```

INTRODUCTION

Features

- ❏ Add Records - Register a new employee with ID, name, position, hourly rate, and hours worked.
- ❏ Edit Records - Modify an existing employee's details.
- ❏ Delete Records - Remove an employee from the system.
- ❏ Compute Salary - Calculate an employee's net salary after deductions.
- ❏ Show All Employees - Display all employee records.
- ❏ Persistent Storage - Save and load employee data from a file (payroll.db).

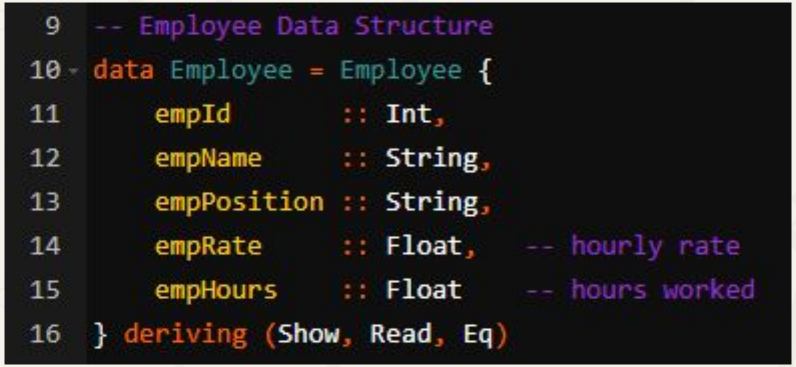


```
=== Simple Payroll System with Persistent Storage ===
Choose an option:
1. Add Employee
2. Edit Employee
3. Delete Employee
4. Compute Salary
5. Show All Employees
6. Exit (Save & Cleanup)
Enter your choice: 1
== Add Employee ==
Enter Employee ID: 1098752
Enter Employee Name: John F. Kenneth Dee
Enter Position: Human Resources
Enter Hourly Rate: 450
Enter Hours Worked: 8
Employee added!
```

CODE STRUCTURE

Employee Data Type

- ❑ Represents an employee with fields: empId, empName, empPosition, empRate, and empHours.
- ❑ Stores employee data in an immutable structure.
- ❑ Encapsulates data by giving controlled access.

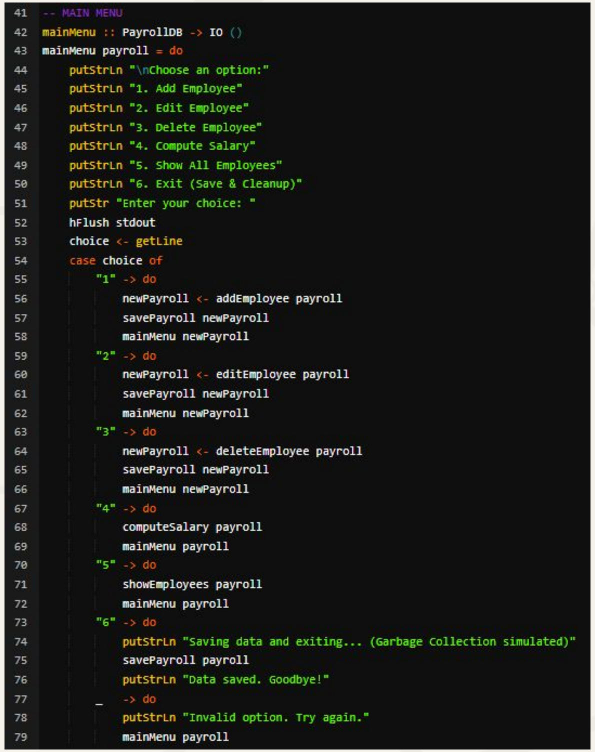


```haskell
-- Employee Data Structure
data Employee = Employee {
    empId       :: Int,
    empName     :: String,
    empPosition :: String,
    empRate     :: Float,   -- hourly rate
    empHours    :: Float    -- hours worked
} deriving (Show, Read, Eq)
```

CODE STRUCTURE

Main Menu Implementation

- ❑ Displays the main menu with options.
- ❑ Performs actions based on user input.
- ❑ Save changes to the file before exiting.



```haskell
-- MAIN MENU
mainMenu :: PayrollDB -> IO ()
mainMenu payroll = do
    putStrLn "\nChoose an option:"
    putStrLn "1. Add Employee"
    putStrLn "2. Edit Employee"
    putStrLn "3. Delete Employee"
    putStrLn "4. Compute Salary"
    putStrLn "5. Show All Employees"
    putStrLn "6. Exit (Save & Cleanup)"
    putStr "Enter your choice: "
    hFlush stdout
    choice <- getLine
    case choice of
        "1" -> do
            newPayroll <- addEmployee payroll
            savePayroll newPayroll
            mainMenu newPayroll
        "2" -> do
            newPayroll <- editEmployee payroll
            savePayroll newPayroll
            mainMenu newPayroll
        "3" -> do
            newPayroll <- deleteEmployee payroll
            savePayroll newPayroll
            mainMenu newPayroll
        "4" -> do
            computeSalary payroll
            mainMenu payroll
        "5" -> do
            showEmployees payroll
            mainMenu payroll
        "6" -> do
            putStrLn "Saving data and exiting... (Garbage Collection simulated)"
            savePayroll payroll
            putStrLn "Data saved. Goodbye!"
        _   -> do
            putStrLn "Invalid option. Try again."
            mainMenu payroll
```

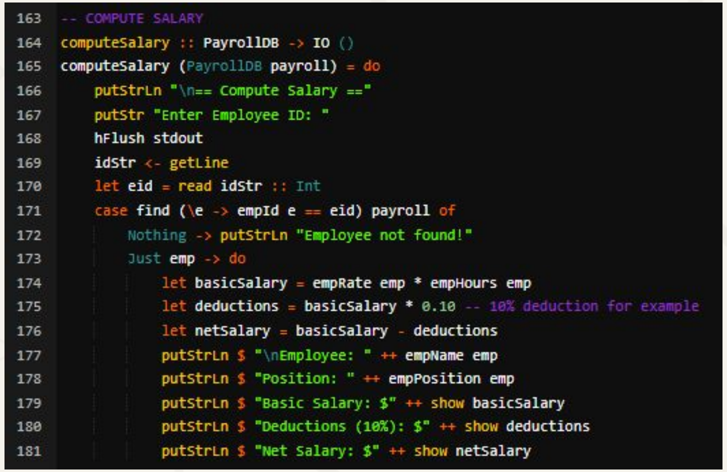
CODE STRUCTURE

Salary Computation

❑ Finds an employee by ID.

Formula

❑ Basic Salary = Hourly Rate × Hours Worked
❑ Deductions = 10% of Basic Salary
❑ Net Salary = Basic Salary - Deductions



```haskell
-- COMPUTE SALARY
computeSalary :: PayrollDB -> IO ()
computeSalary (PayrollDB payroll) = do
    putStrLn "\n== Compute Salary =="
    putStr "Enter Employee ID: "
    hFlush stdout
    idStr <- getLine
    let eid = read idStr :: Int
    case find (\e -> empId e == eid) payroll of
        Nothing -> putStrLn "Employee not found!"
        Just emp -> do
            let basicSalary = empRate emp * empHours emp
            let deductions = basicSalary * 0.10 -- 10% deduction for example
            let netSalary = basicSalary - deductions
            putStrLn $ "\nEmployee: " ++ empName emp
            putStrLn $ "Position: " ++ empPosition emp
            putStrLn $ "Basic Salary: $" ++ show basicSalary
            putStrLn $ "Deductions (10%): $" ++ show deductions
            putStrLn $ "Net Salary: $" ++ show netSalary
```

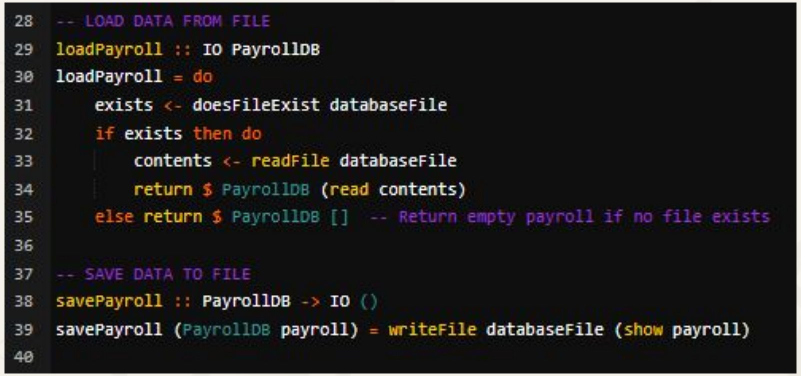
CODE STRUCTURE

Loading Data

- ❑ Checks if the file ‘payroll.db’ exists.
- ❑ If file exists, it reads the contents and converts them into a list of employees.

Saving Data

- ❑ Converts the list of employees into a string representation.
- ❑ Writes the string to payroll.db, ensuring persistent storage.



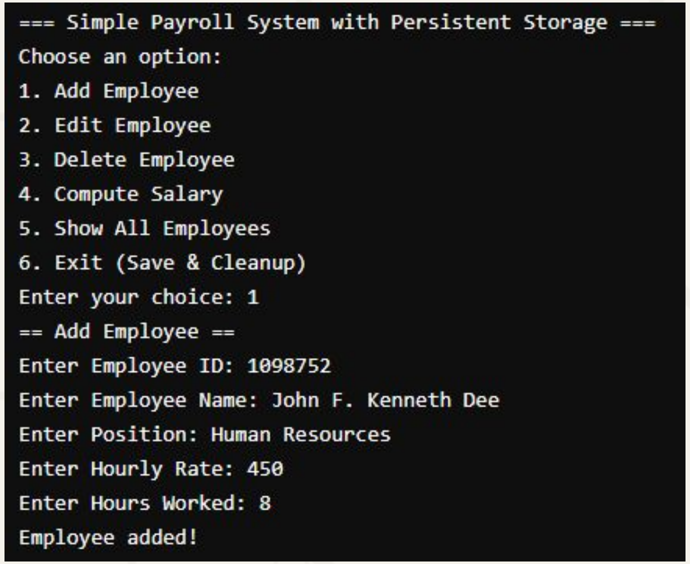
```haskell
-- LOAD DATA FROM FILE
loadPayroll :: IO PayrollDB
loadPayroll = do
    exists <- doesFileExist databaseFile
    if exists then do
        contents <- readFile databaseFile
        return $ PayrollDB (read contents)
    else return $ PayrollDB []  -- Return empty payroll if no file exists

-- SAVE DATA TO FILE
savePayroll :: PayrollDB -> IO ()
savePayroll (PayrollDB payroll) = writeFile databaseFile (show payroll)

```

CONCLUSION

Conclusion

The Haskell program payroll system efficiently manages employee records and computes salaries while ensuring data persistence. It demonstrates fundamental Haskell programming concepts such as:

- ❑ File I/O
- ❑ Data encapsulation



```
=== Simple Payroll System with Persistent Storage ===
Choose an option:
1. Add Employee
2. Edit Employee
3. Delete Employee
4. Compute Salary
5. Show All Employees
6. Exit (Save & Cleanup)
Enter your choice: 1
== Add Employee ==
Enter Employee ID: 1098752
Enter Employee Name: John F. Kenneth Dee
Enter Position: Human Resources
Enter Hourly Rate: 450
Enter Hours Worked: 8
Employee added!
```

THANK YOU!