# EMPLOYEES PAYROLL MANAGEMENT WITH MYSQL INTEGRATION IN PYTHON


Group 6:

**Kheyral Sutan Dumas (2340040301)**

**Charisma Bayu Majestyno (2320010193)**

Faculty:

**Mr. Ivan Firdaus S.T**

Class:

**3CS1**

**CEP CCIT FACULTY OF ENGINEERING**

**UNIVERSITY OF INDONESIA**


**2024**

## PROJECT INFORMATION

**Project Title** **:** Employees Payroll Management With MySQL Integration in Python

**Batch Code** **:** 3CS1

**Start Date** **:** September 20, 2024

**End Date** **:** 1 October, 2024

**Name of Faculty** **:** Mr. Ivan Firdaus S.T

**Names of Developer :**

1. Kheyral Sutan Dumas
2. Charisma Bayu Majestyno

## ACKNOWLEDGEMENT

The author would like to acknowledge the completion of the insightful paper entitled "**Employees Payroll Management With MySQL Integration in Python**." This paper comprehensively discusses the development of a payroll management system using MySQL and Python, focusing on key functions such as employee records, salary calculations, and payroll report generation.

**However, the project is still far from perfect, as it contains numerous bugs, inconsistencies in naming conventions, and areas where functionality can be further refined**. These imperfections reflect the iterative nature of software development and the challenges faced during the integration of complex systems like payroll management.

Overall, the paper serves as a significant contribution to the growing body of knowledge on payroll management systems using MySQL integration in Python applications, while recognizing the need for continued development and refinement.

Depok, 29 September 2024

The Authors

# SYSTEM ANALYSIS

The primary objective is to create a functional system that allows users to manage employee records, salaries records, and generate payroll reports. The system is designed to simplify payroll processing by utilizing Python for application logic and MySQL for reliable database services. Users are able to add, edit, and manage employee data, assign positions, and manage salaries based on defined parameters like position, contract type, and salary deductions.

To enhance the functionality of the system, several Python libraries have been incorporated, such as **mysql-connector-python** and **csv** for the main core library and many more. These libraries help streamline various features, while also supporting better error handling.

However, as with any software project, the system remains imperfect. There may be bugs or unforeseen errors due to inconsistent naming conventions or unhandled edge cases, and the error handling implemented in the current version may not be fully robust, further debugging and testing are necessary to ensure it operates smoothly across all scenarios.

Despite these limitations, the system provides a solid foundation for future improvements.

## PREPARATION

Before commencing with the development process, it is essential to prepare the necessary requirements:

1. Install MySQL Server
https://dev.mysql.com/downloads/installer/

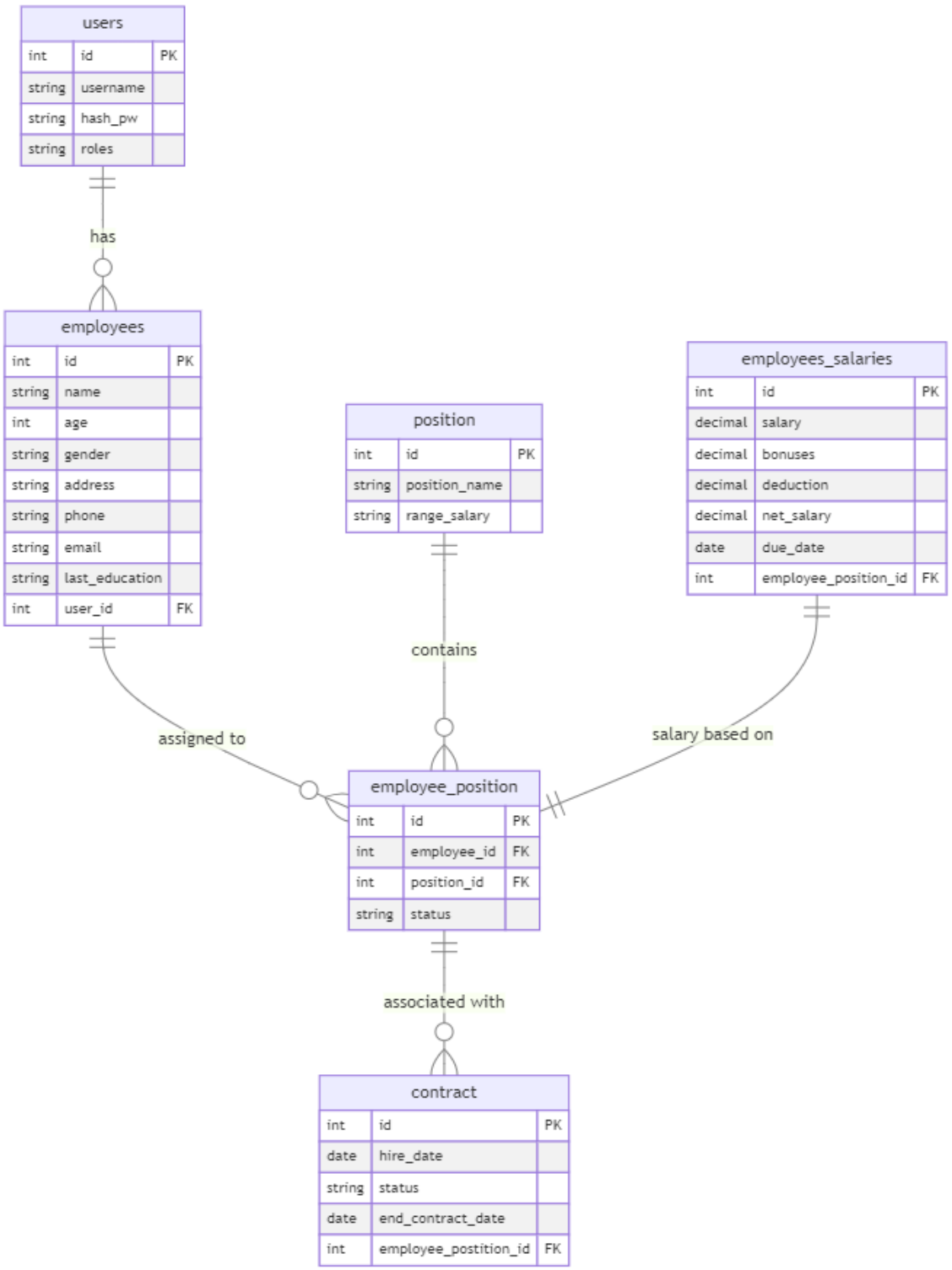
2. Install Python
https://www.python.org/downloads/release/python-3122/

3. Install Visual Studio Codes
https://code.visualstudio.com/download

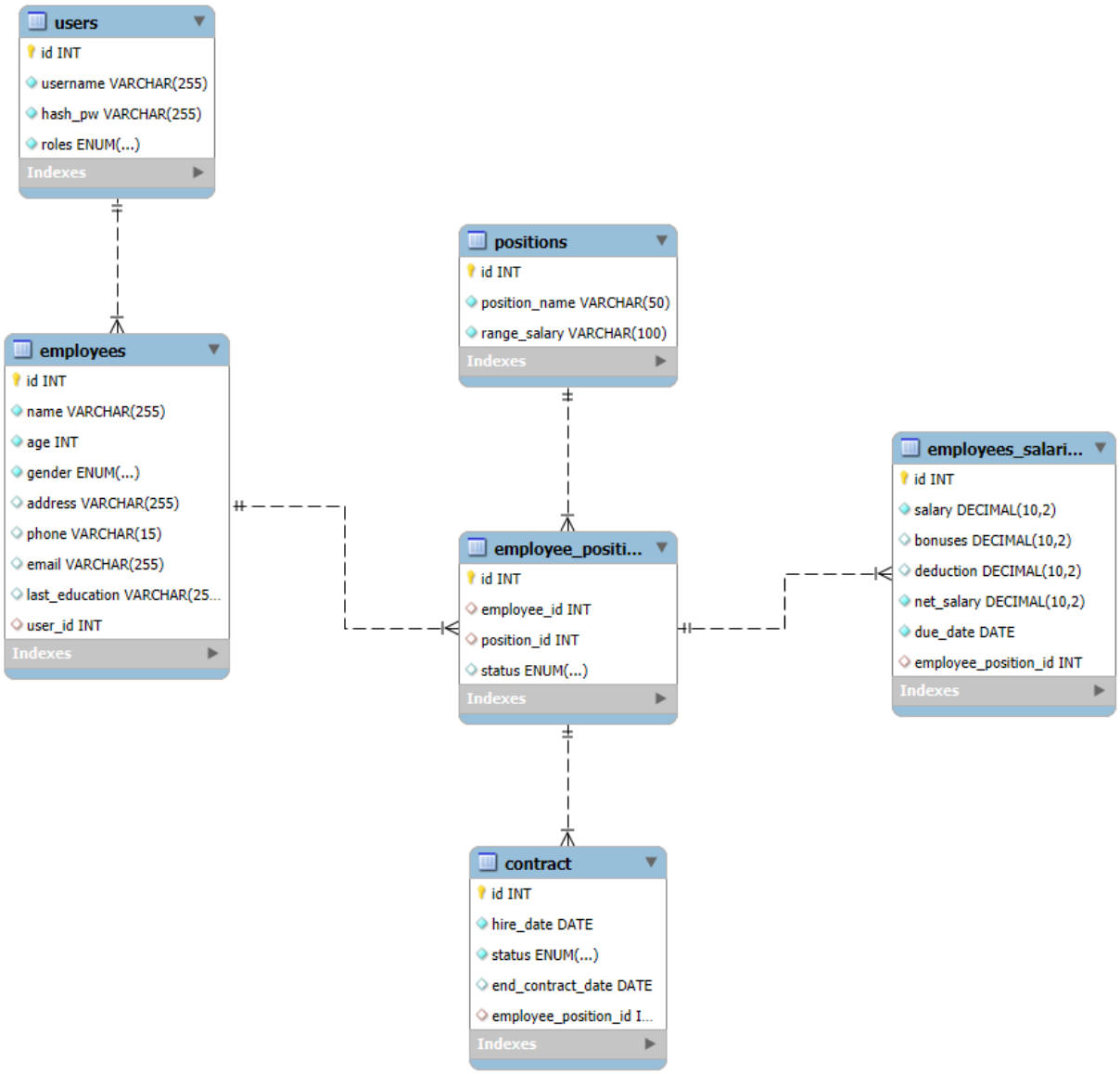
And don't forget to install necessary extension such as IntelliCode, Python, and Etc.

Keep in mind in this paper the authors using Windows 10 as the Operating System to develop the project, Readers can adjust the preparation based on their own Operating System.

# ENTITY RELATIONSHIP DIAGRAM



# SCHEMATIC DIAGRAM



# TABLE DESIGN

## 1. users

| Column Name | Data Type | Size | Key | Delete Rule |
|---|---|---|---|---|
| id (auto_increment) | INT | NULL | PK | NULL |
| username | VARCHAR | 255 | NULL | NULL |
| hash_pw | VARCHAR | 255 | NULL | NULL |
| roles | ENUM | NULL | NULL | NULL |

## 2. employees

| Column Name | Data Type | Size | Key | Delete Rule |
|---|---|---|---|---|
| id (auto_increment) | INT | NULL | PK | NULL |
| name | VARCHAR | 255 | NULL | NULL |
| age | VARCHAR | 255 | NULL | NULL |
| gender | ENUM | NULL | NULL | NULL |
| address | VARCHAR | 255 | NULL | NULL |
| phone | VARCHAR | 30 | NULL | NULL |
| email | VARCHAR | 255 | NULL | NULL |
| last_education | VARCHAR | 255 | NULL | NULL |
| user_id | INT | NULL | FK | ON CASCADE |

## 3. positions

| Column Name | Data Type | Size | Key | Delete Rule |
|---|---|---|---|---|
| id (auto_increment) | INT | NULL | PK | NULL |
| position_name | VARCHAR | 50 | NULL | NULL |
| range_salary | VARCHAR | 100 | NULL | NULL |

# TABLE DESIGN

## 4. employee_position

| Column Name | Data Type | Size | Key | Delete Rule |
|---|---|---|---|---|
| id (auto_increment) | INT | NULL | PK | NULL |
| employee_id | INT | NULL | FK | ON CASCADE |
| range_salary | INT | NULL | FK | NULL |
| status | ENUM | NULL | NULL | NULL |

## 5. employees_salaries

| Column Name | Data Type | Size | Key | Delete Rule |
|---|---|---|---|---|
| id (auto_increment) | INT | NULL | PK | NULL |
| salary | INT | NULL | NULL | NULL |
| bonuses | INT | NULL | NULL | NULL |
| deduction | INT | NULL | NULL | NULL |
| net_salary | INT | NULL | NULL | NULL |
| due_date | DATE | NULL | NULL | NULL |
| employee_position_id | INT | NULL | FK | ON CASCADE |

## 6. contract

| Column Name | Data Type | Size | Key | Delete Rule |
|---|---|---|---|---|
| id (auto_increment) | INT | NULL | PK | NULL |
| hire_date | DATE | NULL | NULL | NULL |
| status | ENUM | NULL | NULL | NULL |
| end_contract_date | DATE | NULL | NULL | NULL |
| employee_position_id | INT | NULL | FK | ON CASCADE |

# CODES

**main.py**

This source code file is the core of the projects, consisting database cursor and connection that will be used across the codes structure, and also containing user input dashboard for performing action.

main.py

```python
from getpass import getpass as hide
import mysql.connector
import account
import manage_employee
import manage_salaries
import view
from generate_csv import generate_payroll_report

# Connection & Cursor setup
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="reds",
    database="payroll"
)

cursor = connection.cursor()
```

# CODES

main.py

```python
# Employee Dashboard
def employee_dashboard(user_id):
    while True:
        print("==================================================")
        print("===        Welcome to Salary Dashboard         ===")
        print("==================================================")
        print("1. View Personal Information")
        print("2. View Payroll Information")
        print("3. Logout")

        option = input("Option: ")

        if option == "1":
            view.view_employees(cursor, specific=True, employee_id=user_id)

        elif option == "2":
            query_position = """
            SELECT ep.id
            FROM employee_position ep
            JOIN employees e ON ep.employee_id = e.id
            WHERE e.user_id = %s
            """
            try:
                cursor.execute(query_position, (user_id,))
                result = cursor.fetchone()
                if result:
                    employee_position_id = result[0]
                    view.view_specific_salary(cursor, employee_position_id)
                else:
                    print("No position record found for this user.")
            except mysql.connector.Error as e:
                print(f"Error: {e}")

        elif option == "3":
            print("Logging out...")
            from time import sleep
            sleep(2)
            break

        else:
            print("Invalid choice! Please select a valid option.")
```

# CODES

main.py

```python
# Admin Dashboard
def admin_dashboard():
    while True:
        print("==================================================")
        print("=== Welcome to Employee Payroll Management 1.0 ===")
        print("==================================================")
        print("1. Manage Employee")
        print("2. Manage Salary & Contract")
        print("3. Generate Payroll Report")
        print("4. Create Account")
        print("5. View Records")
        print("6. Logout")

        choice = input("Option: ")

        if choice == "1":
            menu_employee()

        elif choice == "2":
            menu_salaries_contract()

        elif choice == "3":
            print("=== Generate Payroll Report ===")
            file_path = input("Enter file path (or press Enter to use default): ").strip()

            if file_path:
                generate_payroll_report(cursor, export_path=file_path)
            else:
                generate_payroll_report(cursor)

        elif choice == "4":
            create_account()

```

# CODES

main.py

```python
    elif choice == "5":
        print("== View Records ==")
        print("1. View All Employees Record")
        print("2. View Specific Employee Record")
        print("3. View All Salaries Record")
        print("4. View Specific Salary Record")
        print("5. Back to Menu")

        option = input("Option: ")

        if option == "1":
            view.view_employees(cursor, specific=False)

        elif option == "2":
            employee_id = input("Enter Employee ID: ")
            print("\n")
            view.view_employees(cursor, specific=True, employee_id=employee_id)

        elif option == "3":
            view.view_all_salaries(cursor)

        elif option == "4":
            employee_position_id = input("Enter Employee Position ID: ")
            view.view_specific_salary(cursor, employee_position_id)

        elif option == "5":
            continue

        else:
            print("Invalid option! Please choose a valid record option.")

    elif choice == "6":
        from time import sleep
        print("Logging out...")
        sleep(2)
        break

    else:
        print("Invalid option! Please choose a valid option.")
```

# CODES

main.py

```python
# Function to manage employees
def menu_employee():
    while True:
        print("==================================================")
        print("===            Managing Employees              ===")
        print("==================================================")
        print("1. Add Employees")
        print("2. Edit Employees")
        print("3. Delete Employees")
        print("4. Assign Position")
        print("5. Re-Assign Position")
        print("6. Create New Position")
        print("7. Back to Menu")

        option = input("Option: ")

        if option == "1":
            print("=== Add Employees ===")
            name = input("Enter Name: ")
            age = int(input("Enter Age: "))
            gender = input("Enter Gender (MALE/FEMALE): ").upper()
            address = input("Enter Address: ")
            phone = input("Enter Phone: ")
            email = input("Enter Email: ").lower()
            last_education = input("Enter Last Education: ")
            user_id = input("Enter User ID (Leave it blank if employees do not have account yet): ")

            manage_employee.add_employee(cursor, connection, name, age, gender, address, phone, email, last_education, user_id)

        elif option == "2":
            print("=== Modify Employee ===")
            emp_id = input("Enter Employee ID: ")
            print("Which field do you want to edit?")
            print("1. Name\n2. Age\n3. Gender\n4. Address\n5. Phone\n6. Email\n7. Last Education")
            field = input("Choose a field number: ")
            new_value = input("Enter the new value: ")

            manage_employee.edit_employee(cursor, connection, emp_id, field, new_value)

        elif option == "3":
            print("=== Resign Employee ===")
            emp_id = input("Enter Employee User ID to delete: ")
            manage_employee.delete_employee(cursor, connection, emp_id)
```

# CODES

```python
elif option == "4":
    print("=== Assign Employee ===")
    employee_id = input("Enter Employee ID to assign a position: ")

    query_fetch = "SELECT id, position_name FROM positions"
    cursor.execute(query_fetch,)
    list_position = cursor.fetchall()
    print(f"Position List:")
    print("----------------")
    for position in list_position:
        print(position)

    position_id = input("Enter Position ID: ")
    confirm = input(f"Are you sure you want to assign Employee ID {employee_id} to Position ID {position_id}? (yes/no): ").lower()

    if confirm == "yes":
        manage_employee.add_employee_position(cursor, connection, employee_id, position_id)
    else:
        print("Position assignment canceled.")
        return

elif option == "5":
    print("=== Promote/Demote Employee ===")
    employee_id = input("Enter Employee ID to re-assign position: ")

    query_fetch = "SELECT id, position_name FROM positions"
    cursor.execute(query_fetch,)
    list_position = cursor.fetchall()
    print(f"Position List:")
    print("----------------")
    for position in list_position:
        print(position)

    new_position_id = input("Enter New Position ID: ")
    confirm = input(f"Are you sure you want to re-assign Employee ID {employee_id} to Position ID {new_position_id}? (yes/no): ").lower()

    if confirm == "yes":
        manage_employee.edit_employee_position(cursor, connection, employee_id, new_position_id)
    else:
        print("Position re-assignment canceled.")
        return

elif option == "6":
    print("=== Create New Position ===")
    position_name = input("Enter New Position Name: ")
    range_salary = input("Enter Salary Range (e.g., $50.000-$70.000): ")
    confirm = input("Are you sure you want to create this new position? (yes/no): ").lower()

    if confirm == "yes":
        manage_employee.add_position(cursor, connection, position_name, range_salary)
    else:
        print("Position creation canceled.")
        return

elif option == "7":
    return admin_dashboard()
```

# CODES

main.py

```python
    elif option == "2":
        print("=== Edit Salary Record ===")
        try:
            employee_id = int(input("Enter Employee ID: "))

            view.display_salary_range(cursor, employee_id)

            employee_position_id = int(input("Enter employee position ID to update: "))

            if employee_position_id:
                new_salary = float(input("Enter New Salary: "))
                new_bonuses = (input("Enter New Bonuses (press Enter to keep current): "))
                new_deduction = (input("Enter new deduction (press Enter to keep current): "))

                manage_salaries.edit_salary(cursor, connection, employee_position_id, new_salary, new_bonuses, new_deduction)

            else:
                print("Employee Position ID is required.")

        except ValueError as e:
            print(f"Error: {e}")

    elif option == "3":
        print("=== Delete Salary Record ===")
        try:

            employee_position_id = int(input("Enter employee position ID to remove: "))
            manage_salaries.del_employees_salary(cursor, connection, employee_position_id)
        except ValueError:
            print("Invalid ID input.")

```

# CODES

main.py

```python
    elif option == "4":
        print("=== Add Contract ===")
        try:
            employee_id = int(input("Enter Employee ID: "))

            view.display_salary_range(cursor, employee_id)

            employee_position_id = int(input("Enter employee position ID: "))

            query_check = "SELECT id FROM employee_position WHERE id = %s"
            cursor.execute(query_check, (employee_position_id,))
            is_exist = cursor.fetchone()

            if is_exist:
                pass
            else:
                print("Employee Position ID Does Not Exist")
                return

            hire_date = input("Enter hire date (YYYY-MM-DD): ")
            status = input("Enter contract status (fixed/contracted): ").lower()
            if status not in ["fixed", "contracted"]:
                print("Invalid Option!")
                return

            end_contract_date = None
            if status == "contracted":
                end_contract_date = input("Enter contract end date (YYYY-MM-DD): ")

            manage_salaries.add_contract(cursor, connection, employee_position_id, hire_date, status, end_contract_date)
        except ValueError:
            print("Invalid input.")
```

# CODES

main.py

```python

        elif option == "5":
            print("=== Update/Delete Contract ===")
            action = input("Choose 'update' or 'delete': ").lower()
            if action not in ["update", "delete"]:
                print("Invalid Action!")
            else:
                try:
                    employee_position_id = int(input("Enter Employee Position ID: "))
                    if action == "update":
                        hire_date = input("Re-Enter Hire Date: ")
                        status = input("Enter New Status (fixed/contracted): ").lower()
                        end_contract_date = None
                        if status == "contracted":
                            end_contract_date = input("Enter New End Date (YYYY-MM-DD): ")

                        manage_salaries.manage_contract(cursor, connection, employee_position_id, action, hire_date, status, end_contract_date)

                    elif action == "delete":

                        manage_salaries.manage_contract(cursor, connection, employee_position_id, action)

                except ValueError:
                    print("Invalid Input.")

        elif option == "6":
            break
        else:
              print("Invalid Choice. Please Try Again.")
```

# CODES

main.py

```python
# Function to create a new user account
def create_account():
    print("==================================================")
    print("===            Create New Account              ===")
    print("==================================================")

    username = input("Enter Username: ").lower()
    roles = input("Enter User Roles (admin/employee): ").lower()

    if roles not in ["admin", "employee"]:
        print("Choose Valid Roles!")
        return

    if roles == "admin":
        is_admin = True
    else:
        is_admin = False

    password = hide("Enter Password: ")
    re_enter_pw = hide("Re-Enter Password: ")

    if password != re_enter_pw:
        print("Passwords do not match, please try again!")
    else:
        account.regist_user(cursor, connection, username, password, is_admin)
```

main.py

```python
def main():
    username = input("Enter Username: ")
    password = hide("Enter Password: ")

    account.login_user(cursor, username, password)

if __name__ == "__main__":
    main()
```

# CODES

## validate.py

This source code file is designed to validate user inputs such as date, email, age, and gender. While it is possible that some input fields may not have been anticipated by the developers, As inexperienced developers, we acknowledge our lack of expertise.

validate.py

```python
import re
from datetime import datetime

# Validating user input
def validate_input(type, value):
    if type == "age":
        value = int(value)
        if value <= 0 or value > 100:
            raise ValueError("Invalid age, Please enter proper number!")
        else:
            return value

    if type == "email":
        value = value.lower()
        email_pattern = r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"
        if not re.match(email_pattern, value):
            raise ValueError("Invalid email format!")
        else:
            return value

    elif type == "date":
        try:
            datetime.strptime(value, "%Y-%m-%d")
            return value
        except ValueError as e:
            print(e, "Invalid date format (YYYY-MM-DD).")

    elif type == "gender":
        value = value.upper()
        allowed = ["MALE", "FEMALE"]
        if value not in allowed:
            raise ValueError("GOD created 2 gender MALE or FEMALE!")
        else:
            return value
```

# CODES

**manage_employee.py**

This source code file contains a set of functions designed to manage employee information, including adding, editing, and deleting personal data. It also facilitates the assignment, promotion, or demotion of employee positions, as well as the creation of new position titles for future organizational needs.

manage_employee.py

```python
import mysql.connector
from validate import validate_input

# MANAGE EMPLOYEES

# Adding Employee Record
def add_employee(cursor, connection, name, age, gender, address, phone, email, last_education, user_id=None):
    query = """
    INSERT INTO employees
    (name, age, gender, address, phone, email, last_education, user_id)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
    """

    try:
        email = validate_input("email", email)
        gender = validate_input("gender", gender)

        if user_id == '':
            user_id = None

        cursor.execute(query, (name, age, gender, address, phone, email, last_education, user_id))
        connection.commit()
        print(f"Employee {name} Successfully Added!")
    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

manage_employee.py

```python
# Editing Employee Record
def edit_employee(cursor, connection, id, field, new_value):

    query_check = "SELECT id FROM employees WHERE id = %s"
    cursor.execute(query_check, (id,))
    fetch = cursor.fetchone()

    if not fetch:
        print(f"Employee of ID: {fetch} does not exist!")
        return

    field_map = {
        "1": "name",
        "2": "age",
        "3": "gender",
        "4": "address",
        "5": "phone",
        "6": "email",
        "7": "last_education"
    }

    if field not in field_map:
        print("Invalid Field Options!")
        return

    query = f"UPDATE employees SET {field_map[field]} = %s WHERE id = %s"

    try:
        if field == "3":
            new_value = validate_input("gender", new_value)
        elif field == "6":
            new_value = validate_input("email", new_value)
        elif field == "2":
            new_value = validate_input("age", new_value)

        cursor.execute(query, (new_value, id))
        connection.commit()
        print(f"Employee Information of ID: {id} Successfully Updated!")

    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

manage_employee.py

```python
# Deleting Employee Record
def delete_employee(cursor, connection, id):
    confirmation = input(f"Are you sure you want to delete the Employee with ID {id}? This action cannot be undone! (yes/no): ").lower()

    if confirmation == "yes":
        try:
            query = "DELETE FROM users WHERE id = %s"
            cursor.execute(query, (id,))

            connection.commit()
            print(f"Employee of ID: {id} Successfully Resign!")
        except mysql.connector.Error as e:
            print(e)
    elif confirmation == "no":
        print("Action Canceled.")
        return
    else:
        print("Please Choose (yes/no)!")
        return
```

manage_employee.py

```python
# Adding New Position Name
def add_position(cursor, connection, position_name, range_salary):
    query = "INSERT INTO positions (position_name, range_salary) VALUES (%s, %s)"

    try:
        cursor.execute(query, (position_name, range_salary))
        connection.commit()
        print(f"Position '{position_name}' Successfully Added!")
    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

manage_employee.py

```python
# Assign Employee Position
def add_employee_position(cursor, connection, employee_id, position_id):
    query = "INSERT INTO employee_position (employee_id, position_id, status) VALUES (%s, %s, %s)"
    try:
        cursor.execute("SELECT id FROM positions WHERE id = %s", (position_id,))
        position_exists = cursor.fetchone()

        if position_exists:
            status = input("Enter Employment Status (INTERN/EMPLOYMENT): ").upper()
            if status not in ["INTERN", "EMPLOYMENT"]:
                print("Invalid Status! Please enter either 'INTERN' or 'EMPLOYMENT'.")
                return

            cursor.execute(query, (employee_id, position_id, status))
            connection.commit()
            print(f"Employee ID {employee_id} assigned to Position Number {position_id} with status '{status}'.")
        else:
            print("Error: Position ID does not exist.")

    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

manage_employee.py

```python
# Re-Assign Existing Employee Position
def edit_employee_position(cursor, connection, employee_id, new_position_id):
    query_check_existing = "SELECT * FROM employee_position WHERE employee_id = %s"

    try:
        cursor.execute(query_check_existing, (employee_id,))
        existing_position = cursor.fetchone()

        if not existing_position:
            print(f"Error: Employee ID {employee_id} has no position assigned yet.")
            return

        cursor.execute("SELECT id FROM positions WHERE id = %s", (new_position_id,))
        position_exists = cursor.fetchone()

        if position_exists:
            query_update_position = "UPDATE employee_position SET position_id = %s WHERE employee_id = %s"
            cursor.execute(query_update_position, (new_position_id, employee_id))
            connection.commit()
            print(f"Employee ID {employee_id} successfully re-assigned to Position ID {new_position_id}.")
        else:
            print("Error: Position ID does not exist.")

    except mysql.connector.Error as e:
        print(f"Error: {e}")

```

# CODES

**manage_salaries.py**

This source code file contains a set of functions designed to manage employee salaries, including adding, editing, and deleting personal salaries. It also facilitates the assignment of a contract of an employee, also deleting or updating contract based on company regulations.

manage_salaries.py

```python
import mysql.connector
from validate import validate_input

# MANAGING SALARIES

# Add Salary Record
def add_salary(cursor, connection, salary, bonuses=0, deduction=0, due_date=None, employee_position_id=None):
    try:
        due_date = validate_input("date", due_date)

        salary = int(salary)
        bonuses = int(bonuses)
        deduction = int(deduction)

        gross_salary = salary + bonuses
        deduction_amount = (gross_salary * deduction) / 100
        total_salary = gross_salary - deduction_amount
        print(f"Calculated Net Salary: {total_salary}")

        query_salary = """
        INSERT INTO employees_salaries (salary, bonuses, deduction, net_salary, due_date, employee_position_id)
        VALUES (%s, %s, %s, %s, %s, %s)
        """

        cursor.execute(query_salary, (salary, bonuses, deduction, total_salary, due_date, employee_position_id))
        connection.commit()
        print("Salary record successfully added!")

    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

manage_salaries.py

```python
# Updating Salary Record
def edit_salary(cursor, connection, employee_position_id, new_salary, new_bonuses=None, new_deduction=None):
    query_check = "SELECT bonuses, deduction FROM employees_salaries WHERE employee_position_id = %s"

    try:
        cursor.execute(query_check, (employee_position_id,))
        output = cursor.fetchone()

        if output:
            current_bonuses, current_deduction = output

            bonuses = new_bonuses if new_bonuses else current_bonuses
            deduction = new_deduction if new_deduction else current_deduction

            gross_salary = new_salary + bonuses
            deductions_amount = (gross_salary * deduction) / 100
            net_salary = gross_salary - deductions_amount

            query_update = """
            UPDATE employees_salaries
            SET salary = %s, bonuses = %s, deduction = %s, net_salary = %s
            WHERE employee_position_id = %s
            """

            cursor.execute(query_update, (new_salary, bonuses, deduction, net_salary, employee_position_id))
            connection.commit()
            print(f"Salary for Employee Position ID: {employee_position_id} Updated Successfully!")

        else:
            print(f"No Salary Record Found For Employee Position ID: {employee_position_id}")

    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

manage_salaries.py

```python
# Deleting Salary Record
def del_employees_salary(cursor, connection, employee_position_id):
    query_check = "SELECT id FROM employee_position WHERE id = %s"
    query = "DELETE FROM employees_salaries WHERE employee_position_id = %s"

    try:
        cursor.execute(query_check, (employee_position_id,))
        result = cursor.fetchone()

        if result is not None:
            is_exist = result[0]
            if is_exist:
                cursor.execute(query, (employee_position_id,))
                connection.commit()
                print(f"Salary record for Employee Position ID: {employee_position_id} Successfully Deleted!")
        else:
            print("Employee Position ID Not Exist!")
    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

manage_salaries.py

```python
# MANAGE CONTRACT

# Adding Contract Record
def add_contract(cursor, connection, employee_position_id, hire_date, status, end_contract_date=None):
    hire_date = validate_input("date", hire_date)

    if status == "contracted":
        if not end_contract_date:
            print("Contracted employees must have an end contract date.")
            return
        else:
            end_contract_date = validate_input("date", end_contract_date)
            if not end_contract_date:
                print("Invalid end contract date.")
                return

    try:
        query = """
        INSERT INTO contract (employee_position_id, hire_date, status, end_contract_date)
        VALUES (%s, %s, %s, %s)
        """
        cursor.execute(query, (employee_position_id, hire_date, status, end_contract_date))
        connection.commit()
        print("Contract successfully added!")

    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

manage_salaries.py

```python
# Update/Delete Contract Record
def manage_contract(cursor, connection, employee_position_id, action, hire_date=None, status=None, end_contract_date=None):
    try:
        if action == "update":
            if hire_date:
                hire_date = validate_input("date", hire_date)
            if status == "fixed":
                end_contract_date = None
            elif status == "contracted" and end_contract_date:
                end_contract_date = validate_input("date", end_contract_date)

            query_update = """
            UPDATE contract
            SET hire_date = %s, status = %s, end_contract_date = %s
            WHERE employee_position_id = %s
            """
            cursor.execute(query_update, (hire_date, status, end_contract_date, employee_position_id))
            connection.commit()
            print(f"Contract for Employee Position ID: {employee_position_id} Updated Successfully!")

        elif action == "delete":
            query_delete = "DELETE FROM contract WHERE employee_position_id = %s"
            cursor.execute(query_delete, (employee_position_id,))
            connection.commit()
            print(f"Contract for Employee Position ID: {employee_position_id} Deleted Successfully!")

    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

## account.py

This source code file contains functions for employee login and account creation, with the exception that only users with an admin role are permitted to create employee accounts.

account.py

```python
import bcrypt
import mysql.connector
from main import admin_dashboard, employee_dashboard

# Register for new user
def regist_user(cursor, connection, username, password, is_admin):

    hashed_pw = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())

    if is_admin == True:
        query = "INSERT INTO users (username, hash_pw, roles) VALUES (%s, %s, 'admin')"
    else:
        query = "INSERT INTO users (username, hash_pw, roles) VALUES (%s, %s, 'employee')"

    try:
        cursor.execute(query, (username, hashed_pw))
        connection.commit()

        cursor.execute("SELECT id FROM users WHERE username = %s", (username,))
        rows = cursor.fetchall()

        if rows:
            user_id = rows[0][0]
        else:
            print("Error: User ID not found!")
            return

        link_employee = input("Do you want to link this user to an employee? (yes/no): ").lower()

        if link_employee == "yes":
            employee_id = input("Enter Employee ID to link: ")
            update_query = "UPDATE employees SET user_id = %s WHERE id = %s"
            cursor.execute(update_query, (user_id, employee_id))
            connection.commit()
            print(f"User account successfully linked to Employee ID {employee_id}!")

        print("User Successfully Registered!")
    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

account.py

```python
# Login for user
def login_user(cursor, username, password):

    query = "SELECT id, hash_pw, roles FROM users WHERE username = %s"
    cursor.execute(query, (username,))
    output = cursor.fetchone()

    if output:
        user_id, stored_hash, role = output

        if bcrypt.checkpw(password.encode("utf-8"), stored_hash.encode()):
            if role == "admin":
                print(f"Login Successful as {role}!")
                return admin_dashboard()
            else:
                print(f"Login Successful as {role}!")
                return employee_dashboard(user_id)
        else:
            print("Invalid password or username!")
    else:
        print("Username Invalid or Not Exist!")
        return
```

# CODES

**view.py**

This source code file contains functions that allow employees to view their personal information and salary details. however Admin have extended access, enabling them to view individual employee or all employee records, as well as individual salary details or the complete salary record for all employees.

view.py

```python
import mysql.connector

# VIEWING RECORDS

# View Employees (All or Specific)
def view_employees(cursor, specific=None, employee_id=None):
    if specific == True:
        # View specific employee (no changes)
        query = "SELECT * FROM employees WHERE id = %s OR user_id = %s"
        try:
            cursor.execute(query, (employee_id,employee_id))
            employee = cursor.fetchone()

            if employee:
                print("Employee Details:")
                print("+---------------------------------------------+")
                print(f"ID: {employee[0]}")
                print(f"Name: {employee[1]}")
                print(f"Age: {employee[2]}")
                print(f"Gender: {employee[3]}")
                print(f"Address: {employee[4]}")
                print(f"Phone: {employee[5]}")
                print(f"Email: {employee[6]}")
                print(f"Last Education: {employee[7]}")
                print(f"User ID: {employee[8]}")
                print("+---------------------------------------------+\n")
            else:
                print(f"Employee with ID {employee_id} not found.")
        except mysql.connector.Error as e:
            print(f"Error: {e}")
```

# CODES

view.py

```python
else:
    # View all employees
    query = "SELECT * FROM employees"
    try:
        cursor.execute(query)
        employees = cursor.fetchall()

        if employees:
            for employee in employees:
                print("Employee Details:")
                print(f"ID: {employee[0]}")
                print(f"Name: {employee[1]}")
                print(f"Age: {employee[2]}")
                print(f"Gender: {employee[3]}")
                print(f"Address: {employee[4]}")
                print(f"Phone: {employee[5]}")
                print(f"Email: {employee[6]}")
                print(f"Last Education: {employee[7]}")
                print(f"User ID: {employee[8]}")
                print("------------------------")
        else:
            print("No employees found.")
    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

view.py

```python
# View All Salaries
def view_all_salaries(cursor):
    query = """
    SELECT s.id, s.salary, s.bonuses, s.deduction, s.net_salary, s.due_date, p.position_name
    FROM employees_salaries s
    JOIN positions p ON s.employee_position_id = p.id
    """
    try:
        cursor.execute(query)
        salaries = cursor.fetchall()

        if salaries:
            for salary in salaries:
                due_date_str = salary[5].strftime("%Y-%m-%d")  # Format the due date as YYYY-MM-DD
                print("Salary Details:")
                print(f"ID: {salary[0]}")
                print(f"Salary: {salary[1]}")
                print(f"Bonuses: {salary[2]}")
                print(f"Deduction: {salary[3]}")
                print(f"Net Salary: {salary[4]}")
                print(f"Due Date: {due_date_str}")
                print(f"Position: {salary[6]}")
                print("------------------------")
        else:
            print("No salary records found.")
    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

view.py

```python
# View Specific Salary
def view_specific_salary(cursor, employee_position_id):
    query = """
    SELECT salary, bonuses, deduction, net_salary, due_date
    FROM employees_salaries
    WHERE employee_position_id = %s
    """
    try:
        cursor.execute(query, (employee_position_id,))
        salary_record = cursor.fetchone()

        if salary_record:
            print("Salary Details:")
            print("+-----------------------------------+")
            print(f"Salary: {salary_record[0]}")
            print(f"Bonuses: {salary_record[1]}")
            print(f"Deduction: {salary_record[2]}")
            print(f"Net Salary: {salary_record[3]}")
            print(f"Due Date: {salary_record[4]}")
            print("+-----------------------------------+\n")
        else:
            print(f"No salary record found for Position ID {employee_position_id}.")
    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

view.py

```python
# Display Salaries Range
def display_salary_range(cursor, employee_id):
    query_employee_position_id = "SELECT id FROM employee_position WHERE employee_id = %s"
    query_employee_position = "SELECT position_id FROM employee_position WHERE employee_id = %s"
    query_position = "SELECT position_name, range_salary FROM positions WHERE id = %s"
    query_name = "SELECT name FROM employees WHERE id = %s"

    try:
        cursor.execute(query_employee_position, (employee_id,))
        employee_position = cursor.fetchone()
        if not employee_position:
            print("Error: No position found for this Employee ID.")
            return

        position_id = employee_position[0]

        cursor.execute(query_position, (position_id,))
        position = cursor.fetchone()

        cursor.execute(query_name, (employee_id,))
        name = cursor.fetchone()[0]

        cursor.execute(query_employee_position_id, (employee_id,))
        post_id = cursor.fetchone()[0]

        if position:
            print(f"Name: {name}")
            print(f"Position: {position[0]}")
            print(f"Employee Position ID: {post_id}")
            print(f"Salary Range Reference: {position[1]}")
        else:
            print("Position not found.")
            return

    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

# CODES

**generate.py**

This source code file contains functions that allow admin to export salaries report for the last 30 days.

generate.py

```python
import csv
from datetime import datetime, timedelta
import mysql.connector

# Function to generate payroll report
def generate_payroll_report(cursor, export_path=None):
    if not export_path:
        export_path = r"C:\Users\Jotun\Desktop\payroll_report.csv"

    today = datetime.today()
    last_month = today - timedelta(days=30)

    query = """
    SELECT e.id AS employee_id, e.name, s.salary AS base_salary,
           s.bonuses, s.deduction, s.net_salary,
           p.position_name, c.status AS employment, s.due_date
    FROM employees e
    JOIN employee_position ep ON e.id = ep.employee_id
    JOIN employees_salaries s ON ep.id = s.employee_position_id
    JOIN positions p ON ep.position_id = p.id
    JOIN contract c ON ep.id = c.employee_position_id
    WHERE s.due_date ≥ %s
    """
```

# CODES

generate.py

```python
    try:
        cursor.execute(query, (last_month,))
        payroll_records = cursor.fetchall()

        if not payroll_records:
            print("No payroll records found for the last month.")
            return

        print("+----+-------+-------------+---------+-----------+------------+-----------+-------------+------------------+-----------------+")
        print("| ID | Name  | Base Salary | Bonuses | Deduction | Net Salary | Position  | Employment  | Last Payroll Date | Due Date        |")
        print("+----+-------+-------------+---------+-----------+------------+-----------+-------------+------------------+-----------------+")
        for record in payroll_records:
            print(f"| {record[0]:^2} | {record[1]:^5} | {record[2]:^11} | {record[3]:^7} | {record[4]:^9} | {record[5]:^10} | {record[6]:^12} | {record[7]:^11} | {today.strftime('%Y-%m-%d')} | {record[8]} |")
        print("+----+-------+-------------+---------+-----------+------------+-----------+-------------+------------------+-----------------+")

        with open(export_path, mode='w', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(["Employee ID", "Name", "Base Salary", "Bonuses", "Deduction", "Net Salary", "Position", "Employment", "Last Payroll Date", "Due Date"])
            for record in payroll_records:
                writer.writerow([record[0], record[1], record[2], record[3], record[4], record[5], record[6], record[7], today.strftime('%Y-%m-%d'), record[8]])

        print(f"Payroll report successfully exported to {export_path}")

    except mysql.connector.Error as e:
        print(f"Error: {e}")
```

## DRY RUN

1. User login as an Admin

```
Windows PowerShell
(payroll-management) PS C:\Users\Jotun\Documents\Cod
Enter Username: admin
Enter Password:
Login Successful as admin!
==================================================
=== Welcome to Employee Payroll Management 1.0 ===
==================================================
1. Manage Employee
2. Manage Salary & Contract
3. Generate Payroll Report
4. Create Account
5. View Records
6. Logout
Option:
```

2. Let's say a new employee has been accepted into the company. As an admin user, you'll need to add the new employee's personal information into the system. To do this navigate to the "Manage Employee" > "Add Employee" Option and fill all the necessary details.

```
==================================================
===            Managing Employees              ===
==================================================
1. Add Employees
2. Edit Employees
3. Delete Employees
4. Assign Position
5. Re-Assign Position
6. Create New Position
7. Back to Menu
Option: 1
=== Add Employees ===
Enter Name: Kheyral Sutan Dumas
Enter Age: 19
Enter Gender (MALE/FEMALE): male
Enter Address: H. Suhaemi Main St
Enter Phone: +62 857-7696-8978
Enter Email: kheyralsutan@gmail.com
Enter Last Education: Diploma of Cybersecurity
Enter User ID (Leave it blank if employees do not have account yet):
Employee Kheyral Sutan Dumas Successfully Added!
```

## DRY RUN

3. If the user mistakenly inputs some information about the employee, they can go to the "Edit Employees" option to correct the mistake. Simply input the ID of employee, select the field and then update the incorrect details, and save the changes.

```
Windows PowerShell
=== Modify Employee ===
Enter Employee ID: 6
Which field do you want to edit?
1. Name
2. Age
3. Gender
4. Address
5. Phone
6. Email
7. Last Education
Choose a field number: 1
Enter the new value: Kheyral Sutan
Employee Information of ID: 6 Successfully Updated!
```

4. Next, to assign the new employee to the position they applied for, choose the "Assign Position" option. Input the employee's ID and assign the appropriate position. Additionally, as an admin, you can promote, or demote the employee's position if needed using “Re-Assign” option.

```
Windows PowerShell
=== Assign Employee ===
Enter Employee ID to assign a position: 6
Position List:
----------------
(1, 'President')
(2, 'Vice President')
(3, 'Director')
(4, 'Manager')
(5, 'HR')
Enter Position ID: 3
Are you sure you want to assign Employee ID 6 to Position ID 3? (yes/no): yes
Enter Employment Status (INTERN/EMPLOYMENT): employment
Employee ID 6 assigned to Position Number 3 with status 'EMPLOYMENT'.
```

## DRY RUN

5. As the company grows and new job positions emerge, the user can add new positions to the system as a reference in option “Create New Position”. This allows for flexibility in assigning future employees to roles that align with the evolving needs of the organization.

```
Windows PowerShell
==================================================
===              Managing Employees              ===
==================================================
1. Add Employees
2. Edit Employees
3. Delete Employees
4. Assign Position
5. Re-Assign Position
6. Create New Position
7. Back to Menu
Option: 6
=== Create New Position ===
Enter New Position Name: Security Operation Center
Enter Salary Range (e.g., $50.000-$70.000): $6.000-$10.542
Are you sure you want to create this new position? (yes/no): yes
Position 'Security Operation Center' Successfully Added!
```

## DRY RUN

6. After the employee's personal information is recorded, the next step is to define their salary based on their position. To do this, go to "Manage Salary & Contract" > "Add Salary Record." Input the necessary salary details to complete the process.

```
Windows PowerShell
==================================================
===        Managing Salaries & Contracts        ===
==================================================
1. Add Salary Record
2. Edit Salary Record
3. Delete Salary Record
4. Add Contract
5. Update/Delete Contract
6. Back to Main Menu
Option: 1
=== Add Salary Record ===
Enter Employee ID: 6
Name: Kheyral Sutan
Position: Director
Employee Position ID: 5
Salary Range Reference: $28.150 - $40.327
Enter salary: 39200
Enter bonuses: 6900
Enter deduction (tax): 12
Enter due date (YYYY-MM-DD): 2024-10-1
Calculated Net Salary: 40568.0
Salary record successfully added!
```

7. Admin can also edit an employee's salary record if the employee has been promoted or demoted by selecting the "Edit Salary Record" option. Additionally, users have the ability to delete an existing employee record if the employee is no longer with the company.

## DRY RUN

8. Once the salary is set, next is to create a contract by selecting the “Add Contract” option. The admin can then enter details such as the employee's hire date and specify whether the employee is on a contract or permanent status. If the employee is on a contract, the admin will be prompted to provide the contract end date

```
Windows PowerShell
==================================================
===        Managing Salaries & Contracts        ===
==================================================
1. Add Salary Record
2. Edit Salary Record
3. Delete Salary Record
4. Add Contract
5. Update/Delete Contract
6. Back to Main Menu
Option: 4
=== Add Contract ===
Enter Employee ID: 6
Name: Kheyral Sutan
Position: Director
Employee Position ID: 5
Salary Range Reference: $28.150 - $40.327
Enter employee position ID: 5
Enter hire date (YYYY-MM-DD): 2024-9-30
Enter contract status (fixed/contracted): fixed
Contract successfully added!
```

9. There are also update and delete contract features available for contracted employees. This allows the admin to extend an employee's contract if necessary or delete the contract once it has ended.

## DRY RUN

10. After everything is set, it's time to create an account for the newly joined employee. This account will enable the employee to access their payroll details and personal information. To do this go to “Create Account” in main menu.

```
Windows PowerShell
==================================================
===            Create New Account              ===
==================================================
Enter Username: rall
Enter User Roles (admin/employee): employee
Enter Password:
Re-Enter Password:
Do you want to link this user to an employee? (yes/no): yes
Enter Employee ID to link: 6
User account successfully linked to Employee ID 6!
User Successfully Registered!
```

11. To test the account, log out by selecting the "Logout" option, and then log in as the newly created user to verify the account's functionality.

```
Windows PowerShell
Enter Username: rall
Enter Password:
Login Successful as employee!
==================================================
===        Welcome to Salary Dashboard         ===
==================================================
1. View Personal Information
2. View Payroll Information
3. Logout
Option: 1
Employee Details:
+---------------------------------------------+
ID: 6
Name: Kheyral Sutan
Age: 19
Gender: MALE
Address: H. Suhaemi Main St
Phone: +62 857-7696-8978
Email: kheyralsutan@gmail.com
Last Education: Diploma of Cybersecurity
User ID: 8
+---------------------------------------------+

==================================================
===        Welcome to Salary Dashboard         ===
==================================================
1. View Personal Information
2. View Payroll Information
3. Logout
Option: 2
Salary Details:
+----------------------------------+
Salary: 39200
Bonuses: 6900
Deduction: 12
Net Salary: 40568
Due Date: 2024-10-01
```

## DRY RUN

12. After everything is set, it's time to create an account for the newly joined employee. This account will enable the employee to access their payroll details and personal information. To do this go to “Create Account” in main menu.

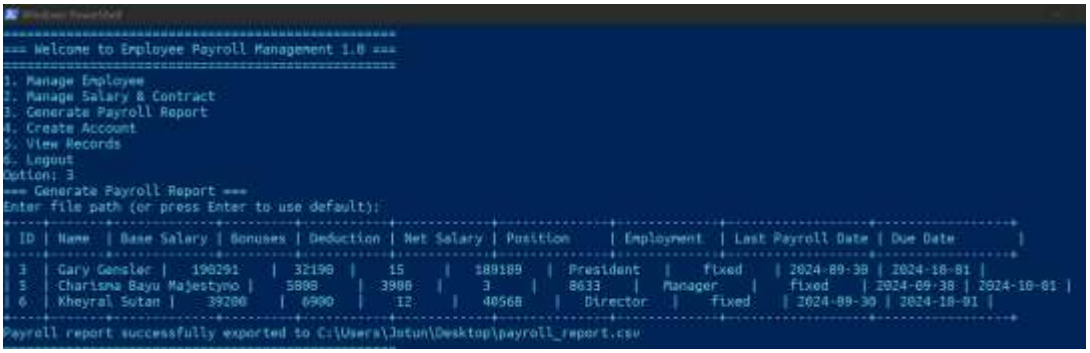
```
Enter Username: rall
Enter Password:
Login Successful as employee!
==================================================
===        Welcome to Salary Dashboard         ===
==================================================
1. View Personal Information
2. View Payroll Information
3. Logout
Option: 1
Employee Details:
+--------------------------------------------+
ID: 6
Name: Kheyral Sutan
Age: 19
Gender: MALE
Address: H. Suhaemi Main St
Phone: +62 857-7696-8978
Email: kheyralsutan@gmail.com
Last Education: Diploma of Cybersecurity
User ID: 8
+--------------------------------------------+

==================================================
===        Welcome to Salary Dashboard         ===
==================================================
1. View Personal Information
2. View Payroll Information
3. Logout
Option: 2
Salary Details:
+----------------------------------+
Salary: 39200
Bonuses: 6900
Deduction: 12
Net Salary: 40568
Due Date: 2024-10-01
```

13. Back login with the admin account, you can access the "View Records" option to either view specific employee information or salary records, or see all employee and salary data at once. While this feature is also available on the employee dashboard, the key distinction is that admins have full visibility of all records, whereas employees can only view their own personal information and salary details.

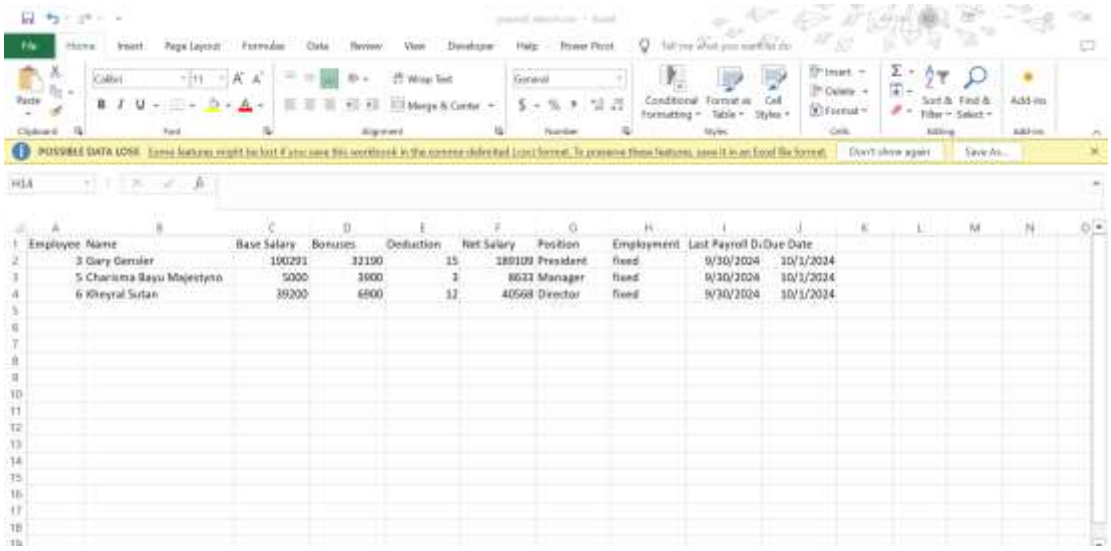
```
=== View Records ===
1. View All Employees Record
2. View Specific Employee Record
3. View All Salaries Record
4. View Specific Salary Record
5. Back to Menu
Option:
```

## DRY RUN

14. After everything is set, it's time to create an account for the newly joined employee. This account will enable the employee to access their payroll details and personal information. To do this go to “Create Account” in main menu.



```
==================================================
=== Welcome to Employee Payroll Management 1.0 ===
==================================================
1. Manage Employee
2. Manage Salary & Contract
3. Generate Payroll Report
4. Create Account
5. View Records
6. Logout
Option: 3
=== Generate Payroll Report ===
Enter file path (or press Enter to use default):
+----+-------+-------------+---------+-----------+-----------+-------------+-----------+-------------------+-----------------+
| ID | Name  | Base Salary | Bonuses | Deduction | Net Salary | Position    | Employment | Last Payroll Date | Due Date        |
+----+-------+-------------+---------+-----------+-----------+-------------+-----------+-------------------+-----------------+
| 3  | Gary Gensler |  190291  |  32190  |    15     |   189189  | President   |   fixed   | 2024-09-30 | 2024-10-01 |
| 5  | Charisma Bayu Majestyno |  5000  |  3900  |    3     |   8633  | Manager   |   fixed   | 2024-09-30 | 2024-10-01 |
| 6  | Kheyral Sutan |  39200  |  6900  |    12     |   40568  | Director   |   fixed   | 2024-09-30 | 2024-10-01 |
+----+-------+-------------+---------+-----------+-----------+-------------+-----------+-------------------+-----------------+
Payroll report successfully exported to C:\Users\Jutun\Desktop\payroll_report.csv
```



| Employee | Name | Base Salary | Bonuses | Deduction | Net Salary | Position | Employment | Last Payroll D | Due Date |
|---|---|---|---|---|---|---|---|---|---|
| 3 | Gary Gensler | 190291 | 32190 | 15 | 189189 | President | fixed | 9/30/2024 | 10/1/2024 |
| 5 | Charisma Bayu Majestyno | 5000 | 3900 | 3 | 8633 | Manager | fixed | 9/30/2024 | 10/1/2024 |
| 6 | Kheyral Sutan | 39200 | 6900 | 12 | 40568 | Director | fixed | 9/30/2024 | 10/1/2024 |

## REQUIREMENTS

**Hardware** :

1. Lenovo V14 G2

**Operating System** :

1. Windows 10 64-bit

**Software** :

1. Visual Studio Code
2. Python
3. MySQL Server

| PROJECT FILE DETAILS | | |
|---|---|---|
| No | Filename | Remarks |
| 1 | 3CS1 Project 1.pdf | Microsoft Words contain documentation paper about the project |
| 2 | sourcecode.py | Files contains the source codes |
| 3 | Project 1 Presentation.pptx | Presentation file |