Payroll Management System Using Python and MySQL

Introduction to Python and MySQL

Python

Python is a user-friendly programming language known for its simplicity and versatility, widely used in web development, automation, and database management.

MySQL

MYSQL is a popular open-source relational database management system (RDBMS) that allows users to store and retrieve data efficiently. It uses SQL (Structured Query Language) to query and manage data. MySQL is widely used in various applications, including web applications and software solutions that require secure and scalable database management.

Introduction to Payroll management system project

The Payroll Management System is developed to simplify the process of managing payroll in an organization. This system provides a user-friendly interface that allows both admins and employees to interact with the payroll data. The system is designed to manage user authentication, employee data, salary calculations, service period tracking, and requests management using Python and MySQL as the backend.

Project Objective

The main objective of this project is to develop a software

solution to automate and simplify payroll processing. The system allows:

- Admins to manage employee data, calculate salaries, and view employee requests.
- Employees to view their salary and service period, withdraw salary, and send requests to the admin.

Project Overview

The payroll management system allows two types of users:

1. Admin
2. Employee

Admins can manage employee data, view requests, calculate salaries, and handle other administrative tasks, while employees can send requests and view their salary and service period.

The core functionalities of this system include:

👉User registration and login with role-based access (admin/employee).

👉**Admin operations:**

▫Adding employees

▫Viewing employees

▫calculating salaries

▫viewing requests

▫deleting employees

▫Exit

👉Employee operations:

▫Sending requests

▫withdrawing salary

▫checking service period

▫Exit

👉Salary calculation based on overtime hours and overtime rates.

Tools and Technologies Used

Programming Language: Python

Database: MySQL

Database Connector: pymysql library

IDE: visual studio code python kernel (3.12.5)

Data Storage: MySQL tables to store users, employees, requests, and salary-related data.

System features

User Authentication

Sign-Up: New users (admins or employees) can create accounts by providing a username, password, and role.

Login: Users can log in by providing correct credentials. Role-based access control is enforced (admin or employee).

Admin Operations

Admins are responsible for the overall management of the system. Admin-specific functionalities include:

1. **Add Employee:** Admins can add employees to the system by creating entries in both the users and employees tables.
2. **View Employees:** The admin can view the list of employees and their details.
3. **Handle Employee Requests:** Admins can view all employee requests (e.g., salary inquiries, leave requests) stored in the requests table.
4. **Calculate Salary:** Admins can calculate an employee's total salary, factoring in overtime pay, and store the results in the employees table.
5. Delete Employee: Admins can delete an employee's record from the system.

Employee Operations

Employees have limited access and are primarily concerned with personal information and requests. Employee-specific functionalities include:

1. Send Requests: Employees can send requests (such as salary inquiries) to the admin.
2. **Withdraw Salary:** Employees can view and withdraw a part or the full amount of their salary after verification.
3. **View Service Period:** Employees can check how long they've been working in the company (service period in months).

Database Design

The system uses the following tables:

1. **Users:** Stores user credentials and roles (admin or employee).

Columns: username, password, role

2. **Employees:** Stores employee details such as salary and service period.

Columns: username, basic_salary, service_period, total_salary

3. **Requests:** Stores employee requests sent to the admin.

Columns: request_id, username, request

Functionalities and Code Flow

Main Workflow

- The system starts with a login screen where users choose between signing up or logging in.
- Based on user roles, the system dynamically switches between admin and employee functionalities.

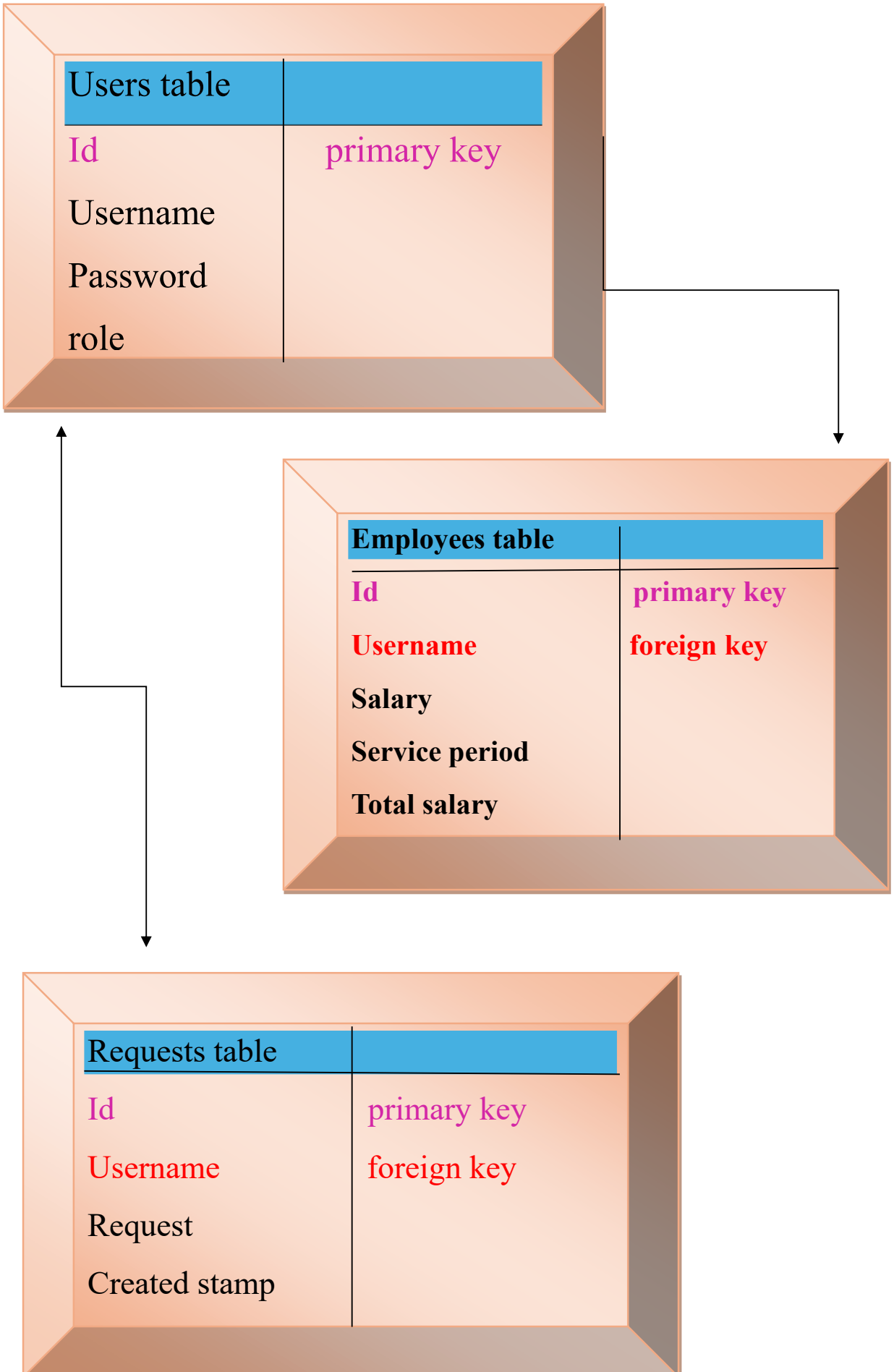
Admin Workflow

Admins can add new employees, calculate their salaries, and manage requests from employees. They can also view or delete employee records.

Employee Workflow

Employees can send requests to the admin, view their service period, and withdraw their salary. The system verifies employee credentials before processing salary withdrawals.

Entity relationship diagram



MYSQL QUERIES FOR CREATING DATABASE AND TABLES

```
CREATE database payroll_management;

USE payroll_management;

CREATE TABLE users (
  Id INT AUTO_INCREMENT PRIMARY KEY,
  Username VARCHAR(255) NOT NULL UNIQUE,
  Password VARCHAR(255) NOT NULL,
  Role ENUM('admin', 'employee') NOT NULL
);

CREATE TABLE employees (
  Id INT AUTO_INCREMENT PRIMARY KEY,
  Username VARCHAR(255) NOT NULL UNIQUE,
  Basic_salary FLOAT NOT NULL,
  Service_period INT NOT NULL,
  FOREIGN KEY (username) REFERENCES users(username)
);
```

```
CREATE TABLE requests (
  Id INT AUTO_INCREMENT PRIMARY KEY,
  Username VARCHAR(255) NOT NULL,
  Request TEXT NOT NULL,
  FOREIGN KEY (username) REFERENCES employees(username)
);
```

Project code

```
Import pymysql

# Connect to the database
Def connect_db():
   Return pymysql.connect(
      Host='localhost',
      User='root',
      Password='root@',
      Database='payroll_'
   )

# Sign-up functionality for new users
Def sign_up():
   Username = input("Enter username: ")
   Password = input("Enter password: ")
   Role = input("Enter role (admin/employee): ").lower()
```

```
    Connection = connect_db()
    Cursor = connection.cursor()

    # Check if username exists and insert user
    Cursor.execute(“SELECT * FROM users WHERE username = %s”, (username,))
    If cursor.fetchone():
        Print(“Username already exists.”)
        Return

    Cursor.execute(“INSERT INTO users (username, password, role) VALUES (%s, %s, %s)”, (username, password, role))
    Connection.commit()
    Print(“Sign-up successful!”)
    Connection.close()

# Login functionality
Def login():
    Username = input(“Enter username: “)
    Password = input(“Enter password: “)
    Role = input(“Enter role (admin/employee): “).lower()

    Connection = connect_db()
    Cursor = connection.cursor()

    # Check user credentials
    Cursor.execute(“SELECT * FROM users WHERE username = %s AND password = %s AND role = %s”, (username, password, role))
```

```
    User = cursor.fetchone()

    If user:
        Print(f"Login successful! Role: {role.capitalize()}")
        If role == 'admin':
            Admin_operations()
        Elif role == 'employee':
            Employee_operations()
        Return True
    Else:
        Print("Invalid username or password.")
        Return False
    Connection.close()

# Admin Operations Menu
Def admin_operations():
    While True:
        Print("\nAdmin Menu")
        Print("1. Add employee")
        Print("2. View employees")
        Print("3. See requests")
        Print("4. Calculate salary of employee")
        Print("5. View service period")
        Print("6. Delete employee")
        Print("7. Exit")

        Choice = input("Enter your choice: ")
```

```
    If choice == ‘1’:
        Add_employee()
    Elif choice == ‘2’:
        View_employees()
    Elif choice == ‘3’:
        See_requests()
    Elif choice == ‘4’:
        Calculate_salary()
    Elif choice == ‘5’:
        View_service_period()
    Elif choice == ‘6’:
        Delete_employee()
    Elif choice == ‘7’:
        Break
    Else:
        Print(“Invalid choice”)

# Employee Operations Menu
Def employee_operations():
  While True:
    Print(“\nEmployee Menu”)
    Print(“1. Send request”)
    Print(“2. Withdraw salary”)
    Print(“3. See my service period”)
    Print(“4. Exit”)

    Choice = input(“Enter your choice: “)
```

```
    If choice == ‘1’:
        Send_request()
    Elif choice == ‘2’:
        Withdraw_salary()
    Elif choice == ‘3’:
        See_service_period()
    Elif choice == ‘4’:
        Break
    Else:
        Print(“Invalid choice”)

# Add a new employee by admin
Def add_employee():
    Username = input(“Enter employee username: “)
    Password = input(“Enter a temporary password for the employee: “)
    Salary = float(input(“Enter basic salary: “))
    Service_period = int(input(“Enter service period (in months): “))

    Connection = connect_db()
    Cursor = connection.cursor()

    # Insert into users and employees tables
    Cursor.execute(“INSERT INTO users (username, password, role) VALUES
(%s, %s, ‘employee’)”, (username, password))
    Connection.commit()

    Cursor.execute(“INSERT INTO employees (username, basic_salary,
service_period, total_salary) VALUES (%s, %s, %s, %s)”,
```

```
            (username, salary, service_period, salary))
    Connection.commit()

    Print("Employee added successfully.")
    Connection.close()

# View all employees
Def view_employees():
    Connection = connect_db()
    Cursor = connection.cursor()

    Cursor.execute("SELECT * FROM employees")
    Employees = cursor.fetchall()
    For emp in employees:
        Print(emp)

    Connection.close()

# See all requests sent by employees
Def see_requests():
    Connection = connect_db()
    Cursor = connection.cursor()

    Cursor.execute("SELECT * FROM requests")
    Requests = cursor.fetchall()
    For req in requests:
        Print(req)
```

```
    Connection.close()

# Calculate employee salary including overtime
Def calculate_salary():
    Username = input("Enter employee username: ")

    Connection = connect_db()
    Cursor = connection.cursor()

    Cursor.execute("SELECT basic_salary FROM employees WHERE username=%s", (username,))
    Employee = cursor.fetchone()

    If employee:
        Basic_salary = employee[0]
        Overtime_hours = int(input("Enter overtime hours worked: "))
        Overtime_rate = float(input("Enter overtime rate: "))

        Hourly_rate = basic_salary / 40
        Overtime_pay = overtime_hours * overtime_rate * hourly_rate
        Total_salary = basic_salary + overtime_pay

        Cursor.execute("UPDATE employees SET total_salary = %s WHERE username = %s", (total_salary, username))
        Connection.commit()

        Print(f"Total salary for {username} updated successfully.")
    Else:
```

```
    Print("Employee not found")

  Connection.close()

# View service period of an employee
Def View_service_period():
  Username = input("Enter employee username: ")

  Connection = connect_db()
  Cursor = connection.cursor()

  Cursor.execute("SELECT service_period FROM employees WHERE
username=%s", (username,))
  Employee = cursor.fetchone()

  If employee:
    Print(f"Service period for {username} is {employee[0]} months")
  Else:
    Print("Employee not found")

  Connection.close()

# Delete an employee from the database
Def delete_employee():
  Username = input("Enter employee username to delete: ")

  Connection = connect_db()
  Cursor = connection.cursor()
```

```
   Cursor.execute("DELETE FROM employees WHERE username=%s",
(username,))
   Connection.commit()

   If cursor.rowcount > 0:
      Print("Employee deleted successfully")
   Else:
      Print("Employee not found")

   Connection.close()

# Send a request by an employee
Def send_request():
   Username = input("Enter your username: ")
   Request = input("Enter your request: ")

   Connection = connect_db()
   Cursor = connection.cursor()

   Cursor.execute("INSERT INTO requests (username, request) VALUES (%s,
%s)", (username, request))
   Connection.commit()
   Print("Request sent successfully")

   Connection.close()

# Withdraw salary by employee and update the balance
```

```
Def withdraw_salary():
   Username = input(“Enter your username: “)

   Connection = connect_db()
   Cursor = connection.cursor()

   Cursor.execute(“SELECT total_salary FROM employees WHERE
username=%s”, (username,))
   Employee = cursor.fetchone()

   If employee:
      Total_salary = employee[0]
      Print(f”Your current total salary is: {total_salary}”)

      Amount_withdraw = float(input(“Enter the amount you wish to withdraw:
“))

      If amount_withdraw > total_salary:
         Print(“Error: Cannot withdraw more than your total salary.”)
      Elif amount_withdraw <= 0:
         Print(“Error: Withdraw amount must be positive.”)
      Else:
         New_salary = total_salary – amount_withdraw
         Print(f”Salary withdrawn: {amount_withdraw}. Remaining balance:
{new_salary}”)

         Cursor.execute(“UPDATE employees SET total_salary = %s WHERE
username = %s”, (new_salary, username))
         Connection.commit()
```

```
        Print("Total salary updated successfully in the database.")
    Else:
        Print("Employee not found.")

    Connection.close()

# See the service period of an employee
Def see_service_period():
    Username = input("Enter your username: ")

    Connection = connect_db()
    Cursor = connection.cursor()

    Cursor.execute("SELECT service_period FROM employees WHERE
username=%s", (username,))
    Employee = cursor.fetchone()

    If employee:
        Print(f"Your service period is {employee[0]} months")
    Else:
        Print("Employee not found")

    Connection.close()

# Main function to handle initial user interaction
Def main():
    While True:
```

```
    Print("\nPayroll Management System")
    Print("1. Sign up")
    Print("2. Login")
    Print("3. Exit")

    Choice = input("Enter your choice: ")

    If choice == '1':
      Sign_up()
    Elif choice == '2':
      Login()
    Elif choice == '3':
      Break
    Else:
      Print("Invalid choice")

If __name__ == "__main__":
  Main()
```

Sample Output of this code:

Initial Program Menu:

Payroll Management System
1. Sign up
2. Login
3. Exit

Enter your choice: 1

Sign Up (New User):

Enter username: john_employee

Enter password: john@123

Enter role (admin/employee): employee

Sign-up successful!

Sign Up (Another New User):

Payroll Management System

1. Sign up
2. Login
3. Exit

Enter your choice: 1

Enter username: alice_admin

Enter password: alice@admin

Enter role (admin/employee): admin

Sign-up successful!

Logging in as Admin:

Payroll Management System

1. Sign up
2. Login
3. Exit

Enter your choice: 2

Enter username: alice_admin

Enter password: alice@admin

Enter role (admin/employee): admin

Login successful! Role: Admin

Admin Menu:

```
Admin Menu
1. Add employee
2. View employees
3. See requests
4. Calculate salary of employee
5. Calculate service period
6. Delete employee
7. Exit
```

Enter your choice: 1

Adding an Employee:

Enter employee username: michael

Enter a temporary password for the employee: michael@456

Enter basic salary: 5000

Enter service period (in months): 12

Employee added successfully.

Viewing Employees:

Admin Menu

1. Add employee
2. View employees
3. See requests
4. Calculate salary of employee
5. Calculate service period
6. Delete employee
7. Exit

Enter your choice: 2

('michael', 5000.0, 12, None) # Output showing username, basic salary, service period, and total salary (None by default)

Logging in as Employee:

Payroll Management System

1. Sign up
2. Login
3. Exit

Enter your choice: 2

Enter username: john_employee

Enter password: john@123

Enter role (admin/employee): employee

Login successful! Role: Employee

Employee Menu:

```
Employee Menu
1. Send request
2. Withdraw salary
3. See my service period
4. Exit
```

Enter your choice: 1

Sending a Request:

Enter your username: john_employee

Enter your request: Need leave approval for 5 days

Request sent successfully

Seeing Service Period:

```
Employee Menu
1. Send request
2. Withdraw salary
3. See my service period
4. Exit
```

Enter your choice: 3

Enter your username: john_employee

Your service period is 12 months

Calculating Salary (Admin):

```
Admin Menu
1. Add employee
2. View employees
3. See requests
4. Calculate salary of employee
5. Calculate service period
6. Delete employee
7. Exit
```

Enter your choice: 4

Enter employee username: michael

Basic Salary: 5000.0

Enter overtime hours worked: 10

Enter overtime rate: 1.5

Total Salary for michael: 5625.0

Total salary for michael updated successfully in the database

---

Withdrawing Salary (Employee):

```
Employee Menu
1. Send request
2. Withdraw salary
3. See my service period
```

Enter your choice: 2

Enter your username: john_employee

Enter your password: john@123

Your total is: 5000.0

Enter the amount you wish to withdraw: 3000

Salary withdrawn: 3000.0

Admin Deleting an Employee:

```
Admin Menu
1. Add employee
2. View employees
3. See requests
4. Calculate salary of employee
5. Calculate service period
6. Delete employee
7. Exit
```

Enter your choice: 6

Enter employee username to delete: john_employee

Employee deleted successfully .

Conclusion

The Payroll Management System provides an easy-to-use platform for managing payroll and employee data. It simplifies tasks for both admins and employees, making payroll management more efficient.