EMPLOYEE MANAGEMENT SYSTEM

GOVIND MAHESHWARI
EMP_ID: 206199

CONTENTS

INTRODUCTION

- The Employee Management System is a Python-based console application integrated with MySQL for efficient management of employee data. The system is divided into two roles: Admin and Employee, with password-based authentication. Admins have full control, allowing them to add, update, delete, and view employee details. They can manage all employee information, including personal details, job positions, and salaries. Employees, on the other hand, can view and edit their own personal information such as phone number, email, and address, and also change their password. This role-based structure ensures data security and task-specific permissions.
- The system is powered by a MySQL database, which stores employee information in an employees table and user credentials in a users table. All actions, including data insertion, updates, and deletions, are handled via SQL queries executed within the Python program using the mysql-connector-python library. Passwords are securely entered using the getpass module, ensuring sensitive data remains protected. The system’s design allows for flexibility and scalability, making it an efficient solution for organizations to manage their workforce details.

SYSTEM WORKFLOW

1. Database Connection:

- Python connects to the MySQL database employee_management.
- Employee and user data are stored in tables (employees and users).

2. User Authentication:

- Users log in with a username and password.
- The system differentiates between Admin and Employee based

3. Admin and Employee Menu:

- After login, Admins can perform various operations like add, update, delete, and view employee records.
- Employees can view and edit their own information and change their password.

ADMIN FUNCTIONALITIES

- Add Employee: Admins can add new employees by entering details such as name, department, salary, and contact info.
- Update Employee: Admins can modify specific employee details by their ID (e.g., changing salary, email, or address).
- Delete Employee: Admins can delete employees from the database based on their employee ID.
- View Employees: Admins can view a specific employee’s details by ID or view all employee data.

EMPLOYEE FUNCTIONALITIES

- View Personal Details: Employees can view their name, department, salary, and other details.
- Edit Personal Details: Employees can update their contact information like phone number, email, or address.
- Change Password: Employees can securely change their password after authenticating their current credentials.

DATABASE STRUCTURE

1. The system uses two key tables:

- employees table: Stores employee information (ID, name, age, department, position, salary, email, phone, address).
- users table: Stores login credentials (username, password, role).

2. Relationships:

- Each employee is associated with a user account for login purposes.

3. Primary Operations:

- Admins manage employee details via the employees table.
- Login and role management are handled via the users table.

SECURITY FEATURES

- Authentication: Each user (Admin or Employee) must log in with a valid username and password, which are verified using MySQL.
- Password Masking: The system uses the getpass module to hide passwords during entry.
- Role-Based Access: Admins and Employees have different access levels. Admins can modify all employee data, while Employees can only access and update their own data.

USER AUTHENTICATION

```python
# User authentication
def authenticate_user(username, password):
    conn = db_connect()
    cursor = conn.cursor()
    query = "SELECT role FROM users WHERE username = %s AND password = %s"
    cursor.execute(query, (username, password))
    result = cursor.fetchone()
    cursor.close()
    conn.close()
    return result[0] if result else None
```

ADMIN FUNCTIONALITIES

```python
# Admin functionalities
def admin_menu():
    while True:
        print("\nAdmin Menu")
        print("1. Add Employee")
        print("2. Update Employee")
        print("3. Delete Employee")
        print("4. View Employee by ID")
        print("5. View All Employees")
        print("6. Logout")
        choice = input("Enter choice: ")

        if choice == '1':
            add_employee()
        elif choice == '2':
            update_employee()
        elif choice == '3':
            delete_employee()
        elif choice == '4':
            view_employee_by_id()
        elif choice == '5':
            view_all_employees()
        elif choice == '6':
            break
        else:
            print("Invalid choice! Please try again.")
```

INSERTING EMPLOYEES

```python
def add_employee():
    conn = db_connect()
    cursor = conn.cursor()
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    gender = input("Enter gender (Male/Female): ")
    department = input("Enter department: ")
    position = input("Enter position: ")
    salary = float(input("Enter salary: "))
    email = input("Enter email: ")
    phone = input("Enter phone number: ")
    address = input("Enter address: ")

    # Insert employee details into employees table
    query = '''INSERT INTO employees
               (name, age, gender, department, position, salary, email, phone, address)
               VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)'''
    cursor.execute(query, (name, age, gender, department, position, salary, email, phone, address))
    emp_id = cursor.lastrowid  # Get the newly added employee ID

    # Create a user account for the employee
    username = input("Set username for employee: ")
    password = getpass.getpass("Set password for employee: ")
    query = '''INSERT INTO users (username, password, role, emp_id)
               VALUES (%s, %s, 'employee', %s)'''
    cursor.execute(query, (username, password, emp_id))
    conn.commit()
    print("Employee and user account added successfully.")
    cursor.close()
    conn.close()
```

UPDATE EMPLOYEES

```python
def update_employee():
    conn = db_connect()
    cursor = conn.cursor()
    emp_id = int(input("Enter employee ID to update: "))
    print("Enter new details (leave blank to keep unchanged):")
    new_name = input("New name: ")
    new_age = input("New age: ")
    new_gender = input("New gender: ")
    new_department = input("New department: ")
    new_position = input("New position: ")
    new_salary = input("New salary: ")
    new_email = input("New email: ")
    new_phone = input("New phone: ")
    new_address = input("New address: ")

    query = "UPDATE employees SET "
    fields = []
    values = []
    if new_name:
        fields.append("name = %s")
        values.append(new_name)
    if new_age:
        fields.append("age = %s")
        values.append(new_age)
    if new_gender:
        fields.append("gender = %s")
        values.append(new_gender)
    if new_department:
        fields.append("department = %s")
        values.append(new_department)
    if new_position:
        fields.append("position = %s")
        values.append(new_position)
    if new_salary:
        fields.append("salary = %s")
        values.append(new_salary)
    if new_email:
        fields.append("email = %s")
        values.append(new_email)
    if new_phone:
        fields.append("phone = %s")
        values.append(new_phone)
    if new_address:
        fields.append("address = %s")
        values.append(new_address)

    if fields:
        query += ', '.join(fields) + " WHERE id = %s"
        values.append(emp_id)
        cursor.execute(query, values)
        conn.commit()
        print("Employee updated successfully.")
    else:
        print("No changes made.")
    cursor.close()
    conn.close()
```

DELETE EMPLOYEES

```python
def delete_employee():
    conn = db_connect()
    cursor = conn.cursor()
    emp_id = int(input("Enter employee ID to delete: "))
    query = "DELETE FROM employees WHERE id = %s"
    cursor.execute(query, (emp_id,))
    conn.commit()

    # Delete associated user account
    query = "DELETE FROM users WHERE emp_id = %s"
    cursor.execute(query, (emp_id,))
    conn.commit()

    print("Employee and user account deleted successfully.")
    cursor.close()
    conn.close()
```

VIEW_EMPLOYEE_BY_ID

```python
def view_employee_by_id():
    conn = db_connect()
    cursor = conn.cursor()
    emp_id = int(input("Enter employee ID to view: "))
    query = "SELECT * FROM employees WHERE id = %s"
    cursor.execute(query, (emp_id,))
    result = cursor.fetchone()
    if result:
        print("Employee details:", result)
    else:
        print("Employee not found.")
    cursor.close()
    conn.close()
```

VIEW_ALL_EMPLOYEES

```
def view_all_employees():
    conn = db_connect()
    cursor = conn.cursor()
    query = "SELECT * FROM employees"
    cursor.execute(query)
    results = cursor.fetchall()
    for row in results:
        print(row)
    cursor.close()
    conn.close()
```

EMPLOYEE FUNCTIONALITIES

```python
# Employee functionalities
def employee_menu(username):
    while True:
        print("\nEmployee Menu")
        print("1. View My Details")
        print("2. Edit My Details")
        print("3. Change My Password")
        print("4. Logout")
        choice = input("Enter choice: ")

        if choice == '1':
            view_my_details(username)
        elif choice == '2':
            edit_my_details(username)
        elif choice == '3':
            change_my_password(username)
        elif choice == '4':
            break
        else:
            print("Invalid choice! Please try again.")
```

VIEW_MY_DETAILS

```
def view_my_details(username):
    conn = db_connect()
    cursor = conn.cursor()
    query = '''SELECT * FROM employees WHERE id =
                (SELECT emp_id FROM users WHERE username = %s)'''
    cursor.execute(query, (username,))
    result = cursor.fetchone()
    if result:
        print("Your details:", result)
    else:
        print("No details found.")
    cursor.close()
    conn.close()
```

EDIT_MY_DETAILS

```
def edit_my_details(username):
    conn = db_connect()
    cursor = conn.cursor()
    query = '''SELECT id FROM employees WHERE id =
                (SELECT emp_id FROM users WHERE username = %s)'''
    cursor.execute(query, (username,))
    emp_id = cursor.fetchone()[0]

    print("Enter new details (leave blank to keep unchanged):")
    new_email = input("New email: ")
    new_phone = input("New phone: ")
    new_address = input("New address: ")

    query = "UPDATE employees SET "
    fields = []
    values = []
    if new_email:
        fields.append("email = %s")
        values.append(new_email)
    if new_phone:
        fields.append("phone = %s")
        values.append(new_phone)
    if new_address:
        fields.append("address = %s")
        values.append(new_address)

    if fields:
        query += ', '.join(fields) + " WHERE id = %s"
        values.append(emp_id)
        cursor.execute(query, values)
        conn.commit()
        print("Details updated successfully.")
    else:
        print("No changes made.")
    cursor.close()
    conn.close()
```

CHANGE EMPLOYEE PASSWORD

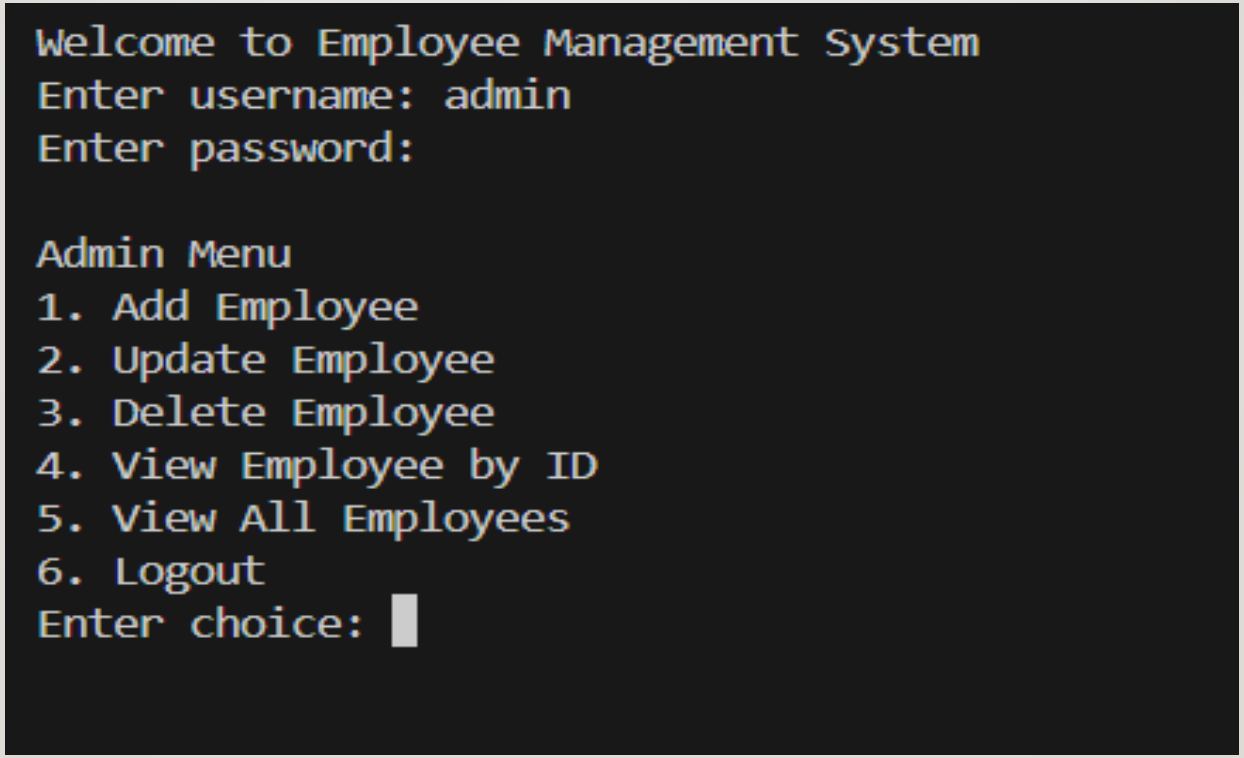
```python
def change_my_password(username):
    conn = db_connect()
    cursor = conn.cursor()

    current_password = getpass.getpass("Enter current password: ")
    query = '''SELECT password FROM users WHERE username = %s'''
    cursor.execute(query, (username,))
    stored_password = cursor.fetchone()[0]

    if current_password == stored_password:
        new_password = getpass.getpass("Enter new password: ")
        confirm_password = getpass.getpass("Confirm new password: ")

        if new_password == confirm_password:
            query = "UPDATE users SET password = %s WHERE username = %s"
            cursor.execute(query, (new_password, username))
            conn.commit()
            print("Password updated successfully.")
        else:
            print("Passwords do not match.")
    else:
        print("Current password is incorrect.")
    cursor.close()
    conn.close()
```

ADMIN MENU OUTPUT



ADDING/INSERTING EMPLOYEE

```
Admin Menu
1. Add Employee
2. Update Employee
3. Delete Employee
4. View Employee by ID
5. View All Employees
6. Logout
Enter choice: 1
Enter name: Rahul
Enter age: 23
Enter gender (Male/Female): Male
Enter department: IT
Enter position: Software Associate
Enter salary: 1000000
Enter email: rahul123@gmail.com
Enter phone number: 123456789
Enter address: 123 Ram Street
Set username for employee: rahul
Set password for employee:
Employee and user account added successfully.
```

UPDATING EMPLOYEE DETIALS

```
Admin Menu
1. Add Employee
2. Update Employee
3. Delete Employee
4. View Employee by ID
5. View All Employees
6. Logout
Enter choice: 2
Enter employee ID to update: 18
Enter new details (leave blank to keep unchanged):
New name:
New age: 24
New gender:
New department:
New position:
New salary: 1200000
New email:
New phone: 2345678910
New address: 123 Elm Street
Employee updated successfully.
```

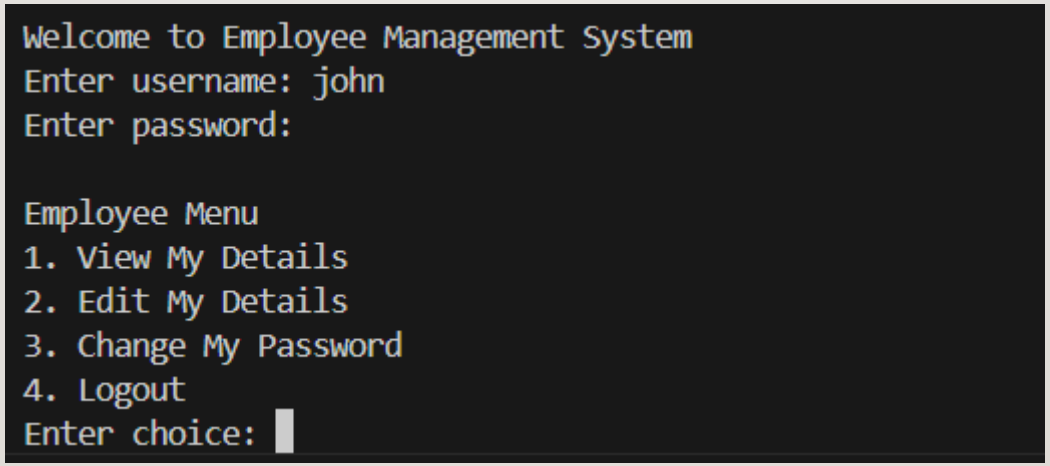
VIEW EMPLOYEE BY ID

```
Admin Menu
1. Add Employee
2. Update Employee
3. Delete Employee
4. View Employee by ID
5. View All Employees
6. Logout
Enter choice: 4
Enter employee ID to view: 18
Employee details: (18, 'Rahul', 24, 'Male', 'IT', 'Software Associate', Decimal('1200000.00'), 'rahul123@gmail.com', '2345678910', '123 Elm Street')
```

DELETE EMPLOYEE

```
Admin Menu
1. Add Employee
2. Update Employee
3. Delete Employee
4. View Employee by ID
5. View All Employees
6. Logout
Enter choice: 3
Enter employee ID to delete: 18
Employee and user account deleted successfully.
```

EMPLOYEE MENU



VIEW EMPLOYEE DETAILS

```
Employee Menu
1. View My Details
2. Edit My Details
3. Change My Password
4. Logout
Enter choice: 1
Your details: (1, 'John', 30, 'Male', 'IT', 'Developer', Decimal('700000.00'), 'john234@gmail.com', '1234567890', '123 Elm Street')
```

THANK YOU!