A
Project Report
On
Employees Management System

by

Jitendra Kumar Dubey

(206612)

Under the guidance

Of

Sachin Vasav

amdocs | make it amazing

Introduction

The Employee Management System (EMS) is a comprehensive software application designed to streamline and automate various employee-related tasks within an organization. This project aims to simplify the management of employee data, from recruitment and onboarding to performance tracking, salary management, and employee exits.

Managing employees is one of the most critical aspects of any business, as employees are the backbone of an organization's productivity and growth. As businesses grow, handling employee records and tracking performance manually becomes cumbersome and prone to errors. Therefore, the need for an efficient, automated system becomes crucial.

This project focuses on providing a user-friendly interface for both administrators and HR personnel to manage essential employee details like personal information, employment history, attendance, salary records, and performance reviews. Additionally, the system will ensure compliance with labour regulations by storing accurate and up-to-date employee information.

The Employee Management System is built using Python for the backend and MySQL for database management, ensuring a robust, scalable, and secure solution for managing employee records. It offers key functionalities such as user registration, login authentication, employee data management, and role-based access, making it a comprehensive tool for organizations of all sizes.

This project not only addresses the challenges of managing a growing workforce but also enhances overall organizational efficiency by enabling real-time access to employee information, reducing paperwork, and improving decision-making processes related to human resources.

Importance of Employee Management:

- **Increases Productivity**: Proper employee management ensures that employees are focused, efficient, and working towards organizational goals.
- **Reduces Turnover**: Engaged and satisfied employees are less likely to leave, reducing recruitment costs and retaining institutional knowledge.
- **Improves Workplace Culture**: Employee management fosters a culture of respect, teamwork, and collaboration, creating a positive and motivating environment.
- **Ensures Compliance**: Managing employees in line with legal requirements helps to avoid fines and legal disputes.
- **Drives Business Growth**: When employees are properly managed, they are more likely to innovate and contribute to the success of the company.

Many companies use Employee Management Systems (EMS) or Human Resource Management Systems (HRMS), which are software solutions to help automate and streamline many of these processes. These systems typically include features for payroll, employee data management, performance tracking, and more.

Effective employee management leads to better employee satisfaction, enhanced productivity, and the overall success of the organization.

Code Snippets

WorkFile.py × User.py × Employee.py ×

```python
import mysql.connector
dataBase= mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="Jit@12345",
    database="employee_management"
)
cursorObject = dataBase.cursor()
# Creation of user table
userRecord="""" CREATE TABLE users (
   id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) UNIQUE,
    password VARCHAR(255)
)"""
cursorObject.execute(userRecord);
dataBase.close()
```

WorkFile.py × User.py × Employee.py ×

```python
import mysql.connector
dataBase =mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="Jit@12345"
)
cursorObject = dataBase.cursor()
# employee_management database creation
cursorObject.execute("CREATE DATABASE employee_management")
dataBase.close()
dataBase= mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="Jit@12345",
    database="employee_management"
)
cursorObject = dataBase.cursor()
# creating a employee table
employeeRecord="""" CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    age INT,
    department VARCHAR(255),
    salary DECIMAL(10, 2)
)"""
cursorObject.execute(employeeRecord);
dataBase.close()
```

WorkFile.py × User.py × Employee.py ×

```python
import mysql.connector
import bcrypt

# making a function to connect database
def connect_to_db():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="Jit@12345",
        database="employee_management"
    )

# hashing
def hash_password(password):
    return bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())

# for verifying  password
def verify_password(stored_password, provided_password):
    return bcrypt.checkpw(provided_password.encode('utf-8'), stored_password.encode('utf-8'))

# Check user is exist in database or not
def user_exists(username):
    conn = connect_to_db()
    cursor = conn.cursor()
    query = "SELECT * FROM users WHERE username = %s"
    cursor.execute(query, (username,))
```

WorkFile.py × User.py × Employee.py ×

```python
    cursor.execute(query, (username,))
    result = cursor.fetchone()
    cursor.close()
    conn.close()
    return result is not None
    # or
    # if result:
    #     return True
    # else:
    #     return False

# Function to register a new employee user
def register_employee_user():
    username = input("Enter a new username: ")
    if user_exists(username):
        print("Username already exists. Try logging in.")
        return False
    password = input("Enter a new password: ")
    hashed_password = hash_password(password)
    conn = connect_to_db()
    cursor = conn.cursor()

    # Insert the new user into the users table
    query = "INSERT INTO users (username, password) VALUES (%s, %s)"
    cursor.execute(query, (username, hashed_password.decode('utf-8')))
    conn.commit()
```

WorkFile.py × User.py × Employee.py ×

```python
    print("User registered successfully!")
    # Once it registered, then we will add Employee details
    name = input("Enter employee name: ")
    age = int(input("Enter employee age: "))
    department = input("Enter employee department: ")
    salary = float(input("Enter employee salary: "))

    query = "INSERT INTO employees (name, age, department, salary) VALUES (%s, %s, %s, %s)"
    cursor.execute(query, (name, age, department, salary))
    conn.commit()
    print("Employee registered successfully!")
    cursor.close()
    conn.close()
    return True


# Function to log in the user
def login():
    username = input("Enter username: ")
    password = input("Enter password: ")

    conn = connect_to_db()
    cursor = conn.cursor()

    query = "SELECT password FROM users WHERE username = %s"
    cursor.execute(query, (username,))
```

WorkFile.py × User.py × Employee.py ×

```python
    cursor.execute(query, (username,))
    # Here we are fetching the password
    result = cursor.fetchone()

    cursor.close()
    conn.close()

    if result:
        stored_password = result[0]
        if verify_password(stored_password, password):
            print("Login successful!")
            return True
        else:
            print("Incorrect password. Try again.")
            return False
    else:
        print("Username not found. Would you like to register as a new employee?")
        register_choice = input("Enter 'yes' to register or 'no' to cancel: ").lower()
        if register_choice == 'yes':
            return register_employee_user()
        else:
            return False


# Function to add a new employee (for general purposes, not the login/registration part)
def add_employee():
```

WorkFile.py × User.py × Employee.py ×

```python
# Function to add a new employee (for general purposes, not the login/registration part)
def add_employee():
    name = input("Enter employee name: ")
    age = int(input("Enter employee age: "))
    department = input("Enter employee department: ")
    salary = float(input("Enter employee salary: "))

    conn = connect_to_db()
    cursor = conn.cursor()

    query = "INSERT INTO employees (name, age, department, salary) VALUES (%s, %s, %s, %s)"
    cursor.execute(query, (name, age, department, salary))

    conn.commit()
    print("Employee added successfully!")
    cursor.close()
    conn.close()


# Function to view all employees
def view_employees():
    conn = connect_to_db()
    cursor = conn.cursor()

    query = "SELECT * FROM employees"
    cursor.execute(query)
```

WorkFile.py × User.py × Employee.py ×

```python
    cursor.execute(query)
    result = cursor.fetchall()

    if result:
        print("\nEmployee List:")
        for row in result:
            print(f"ID: {row[0]}, Name: {row[1]}, Age: {row[2]}, Department: {row[3]}, Salary: {row[4]}")
    else:
        print("No employees found.")

    cursor.close()
    conn.close()


# Function to update employee details
def update_employee():
    emp_id = int(input("Enter employee ID to update: "))

    print("What do you want to update?")
    print("1. Name")
    print("2. Age")
    print("3. Department")
    print("4. Salary")

    choice = int(input("Enter choice: "))

    conn = connect_to_db()
```

user_exists()

WorkFile.py × User.py × Employee.py ×

```python
    conn = connect_to_db()
    cursor = conn.cursor()

    if choice == 1:
        new_name = input("Enter new name: ")
        query = "UPDATE employees SET name = %s WHERE id = %s"
        cursor.execute(query, (new_name, emp_id))
    elif choice == 2:
        new_age = int(input("Enter new age: "))
        query = "UPDATE employees SET age = %s WHERE id = %s"
        cursor.execute(query, (new_age, emp_id))
    elif choice == 3:
        new_department = input("Enter new department: ")
        query = "UPDATE employees SET department = %s WHERE id = %s"
        cursor.execute(query, (new_department, emp_id))
    elif choice == 4:
        new_salary = float(input("Enter new salary: "))
        query = "UPDATE employees SET salary = %s WHERE id = %s"
        cursor.execute(query, (new_salary, emp_id))
    else:
        print("Invalid choice.")
        return

    conn.commit()
    print("Employee updated successfully!")
    cursor.close()
    conn.close()
```

WorkFile.py × User.py × Employee.py ×

```python
    cursor.close()
    conn.close()

# Function to delete an employee
def delete_employee():
    emp_id = int(input("Enter employee ID to delete: "))

    conn = connect_to_db()
    cursor = conn.cursor()

    query = "DELETE FROM employees WHERE id = %s"
    cursor.execute(query, (emp_id,))

    conn.commit()
    print("Employee deleted successfully!")
    cursor.close()
    conn.close()


# Function to search for an employee by ID or name
def search_employee():
    print("Search by:")
    print("1. ID")
    print("2. Name")

    choice = int(input("Enter choice: "))

```

WorkFile.py × User.py × Employee.py ×

```python

    choice = int(input("Enter choice: "))

    conn = connect_to_db()
    cursor = conn.cursor()

    if choice == 1:
        emp_id = int(input("Enter employee ID: "))
        query = "SELECT * FROM employees WHERE id = %s"
        cursor.execute(query, (emp_id,))
    elif choice == 2:
        emp_name = input("Enter employee name: ")
        query = "SELECT * FROM employees WHERE name LIKE %s"
        cursor.execute(query, (f"%{emp_name}%",))
    else:
        print("Invalid choice.")
        return

    result = cursor.fetchall()

    if result:
        for row in result:
            print(f"ID: {row[0]}, Name: {row[1]}, Age: {row[2]}, Department: {row[3]}, Salary: {row[4]}")
    else:
        print("Employee not found.")

    cursor.close()
```

WorkFile.py × User.py × Employee.py ×

```python
    cursor.close()
    conn.close()


# Menu
def main_menu():
    while True:
        print("\nEmployee Management System")
        print("1. Add New Employee")
        print("2. View Employees")
        print("3. Update Employee")
        print("4. Delete Employee")
        print("5. Search Employee")
        print("6. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            add_employee()
        elif choice == 2:
            view_employees()
        elif choice == 3:
            update_employee()
        elif choice == 4:
            delete_employee()
        elif choice == 5:
            search_employee()
```

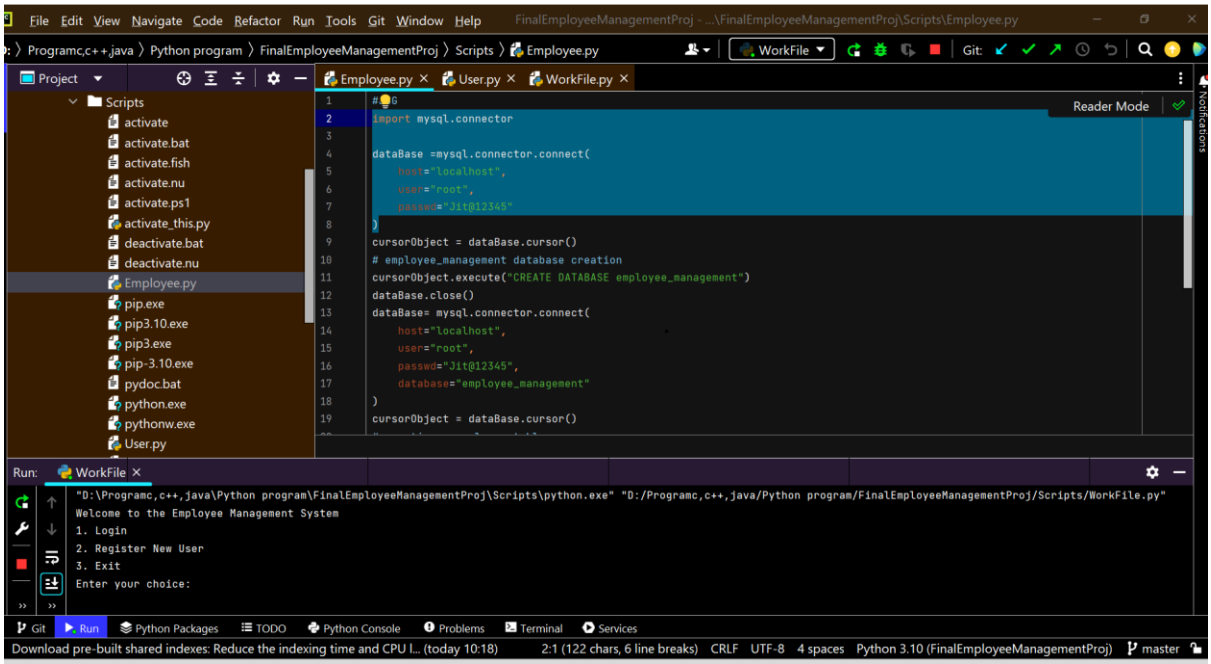
WorkFile.py × User.py × Employee.py ×

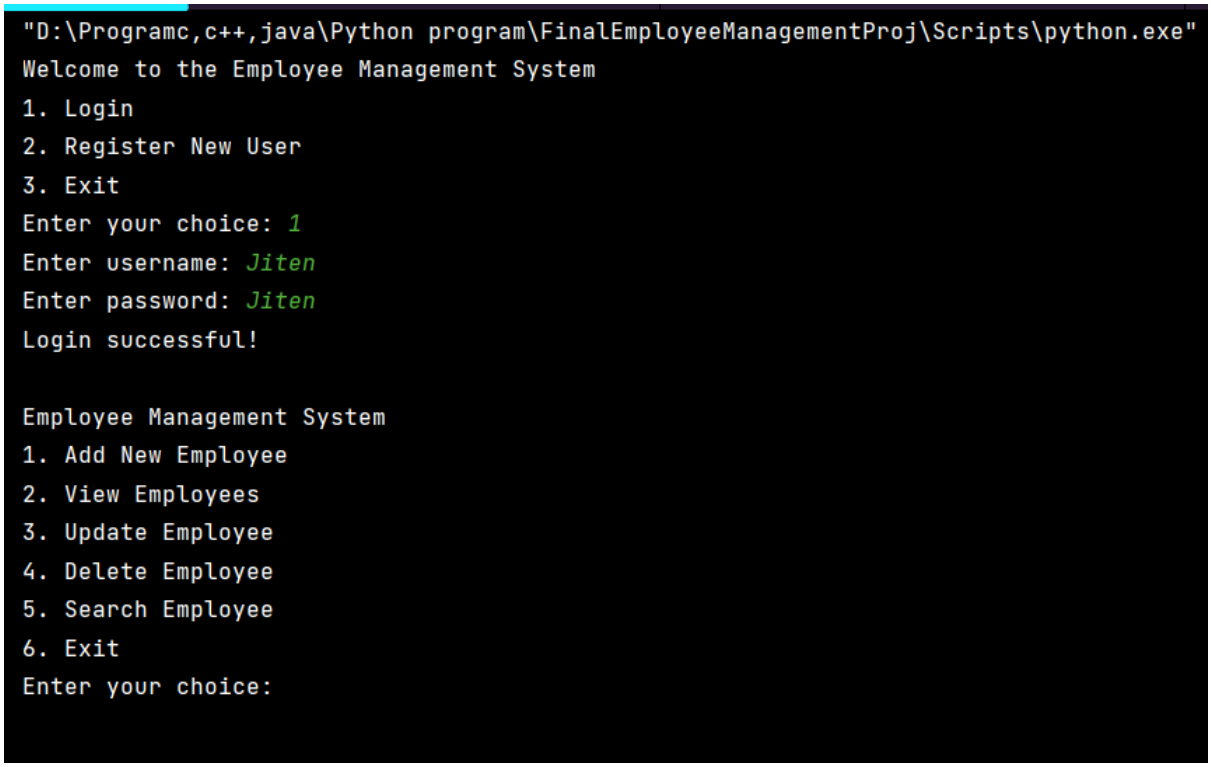
```python
            search_employee()
        elif choice == 6:
            print("Exiting program....")
            break
        else:
            print("Invalid choice so please choose again.")


# Code for start the program
# if __name__ == "__main__":
print("Welcome to the Employee Management System")
while True:
        print("1. Login")
        print("2. Register New User")
        print("3. Exit")
        choice = int(input("Enter your choice: "))

        if choice == 1:
            if login():
                main_menu()
        elif choice == 2:
            register_employee_user()
        elif choice == 3:
            print("Exiting the program.")
            break
        else:
            print("Invalid choice! Please try again.")
```

Output snippets





```
Employee Management System
1. Add New Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Search Employee
6. Exit
Enter your choice: 1
Enter employee name: Suraj
Enter employee age: 24
Enter employee department: MCA
Enter employee salary: 25000
Employee added successfully!

Employee Management System
1. Add New Employee
2. View Employees
3. Update Employee
4. Delete Employee
```

```
6. Exit
Enter your choice: 2

Employee List:
ID: 1, Name: Jiten, Age: 23, Department: IT, Salary: 2000000.00
ID: 2, Name: Anshul, Age: 23, Department: IT, Salary: 85000.00
ID: 3, Name: Kapil, Age: 24, Department: CS, Salary: 54000.00
ID: 5, Name: abc, Age: 21, Department: jk, Salary: 7899.00
ID: 6, Name: mansi, Age: 23, Department: IT, Salary: 70000.00
ID: 7, Name: jittu, Age: 23, Department: iy, Salary: 1000000.00
ID: 8, Name: Rohit, Age: 24, Department: HR, Salary: 450000.00
ID: 10, Name: Suraj, Age: 24, Department: MCA, Salary: 25000.00

Employee Management System
1. Add New Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Search Employee
6. Exit
Enter your choice:
```

Run | Python Packages | TODO | Python Console | Problems | Terminal | Services

```
Employee Management System
1. Add New Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Search Employee
6. Exit
Enter your choice: 3
Enter employee ID to update: 10
What do you want to update?
1. Name
2. Age
3. Department
4. Salary
Enter choice: 4
Enter new salary: 50000
Employee updated successfully!

Employee Management System
1. Add New Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Search Employee
6. Exit
Enter your choice:
```

```
Employee Management System
1. Add New Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Search Employee
6. Exit
Enter your choice: 4
Enter employee ID to delete: 5
Employee deleted successfully!

Employee Management System
1. Add New Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Search Employee
6. Exit
Enter your choice:
```

Git | Run | Python Packages | TODO | Python Console | Problems | Terminal | Services

```
Employee Management System
1. Add New Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Search Employee
6. Exit
Enter your choice: 5
Search by:
1. ID
2. Name
Enter choice: 1
Enter employee ID: 8
ID: 8, Name: Rohit, Age: 24, Department: HR, Salary: 450000.00
```

```
Employee Management System
1. Add New Employee
2. View Employees
3. Update Employee
4. Delete Employee
5. Search Employee
6. Exit
Enter your choice: 5
Search by:
1. ID
2. Name
Enter choice: 2
Enter employee name: Rohit
ID: 8, Name: Rohit, Age: 24, Department: HR, Salary: 450000.00
```

```
Enter your choice: 6
Exiting program....
1. Login
2. Register New User
3. Exit
Enter your choice: 2
Enter a new username: NewUSer
Enter a new password: new123
User registered successfully!
Enter employee name: NewUser
Enter employee age: 43
Enter employee department: Management
Enter employee salary: 40000
Employee registered successfully!
1. Login
2. Register New User
3. Exit
```