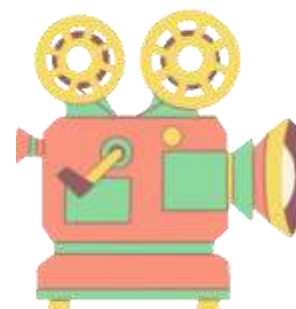




Employee Performance Management System

Enhancing Employee Management & Performance Evaluation

Archana Kalathiya
Employee ID : 207008
Technology Integration Engineer





Agenda

1.

Introduction

2.

Objectives

3.

**Software Design &
Database Structure**

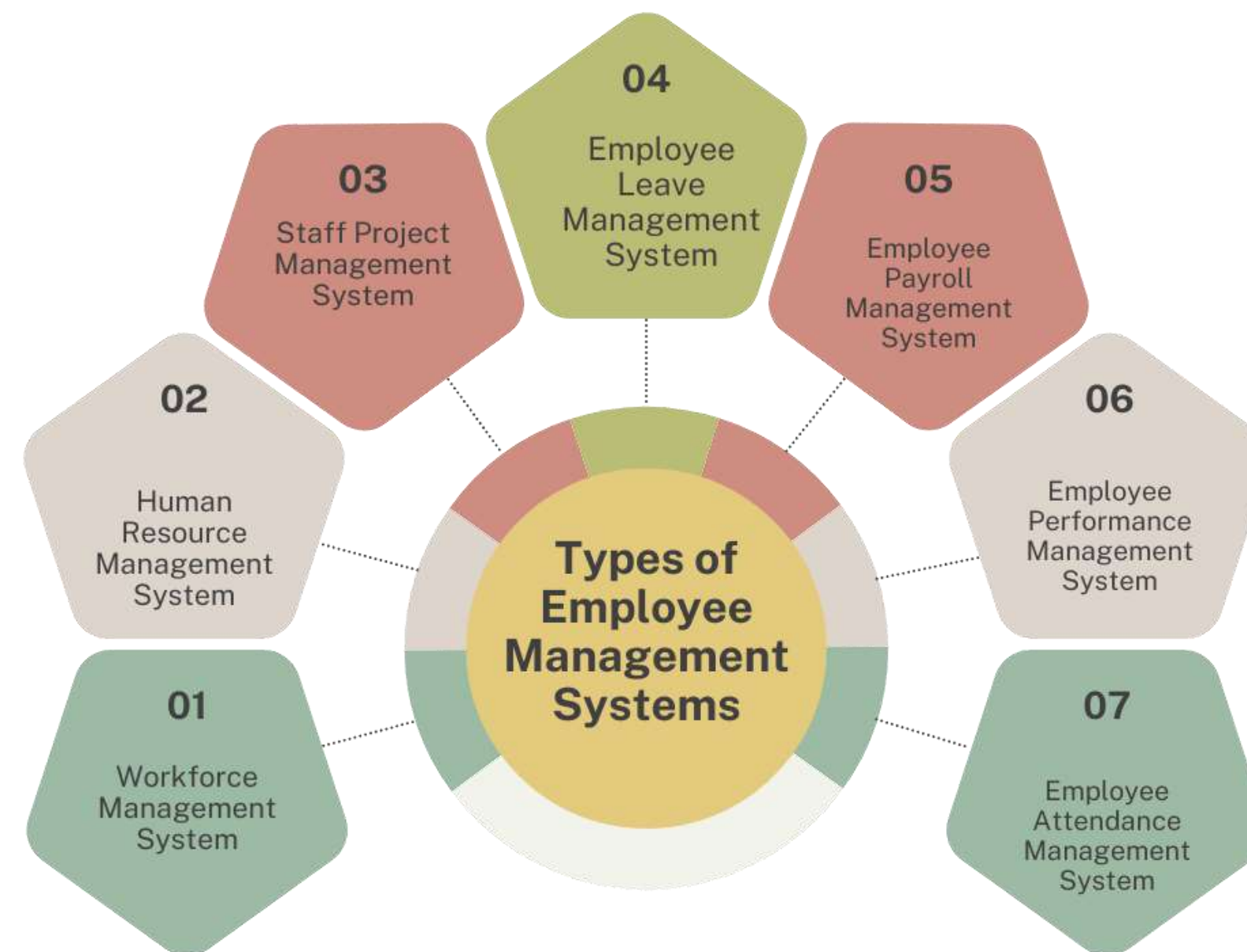
4.

Key Functionalities

5.

Code

To start, let's briefly cover Employee Management Systems (EMS) and their types. While EMS includes various systems for managing general employee information, today, we're focusing on Employee Performance Management Systems (EPMS), which emphasizes tracking and evaluating employee performance to support HR decisions.



Introduction



What is EPMS ?

A system designed to manage employee information, performance, evaluations, and HR processes.

Includes functionalities for employee registration, management, performance, review, and promotion.



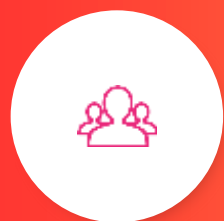
EPMS Objectives

Simplify employee management processes.

Facilitate performance evaluations and structured feedback.

Assist HR in decision-making based on performance data and reviews.





Database layer

- Stores user, employee, and performance review information.
- Connects to MySQL for data persistence.



Business Logic layer

- Functions perform CRUD operations for users, employees, and performance reviews.
- Includes functions for user authentication, registration, and promotions.



Presentation layer

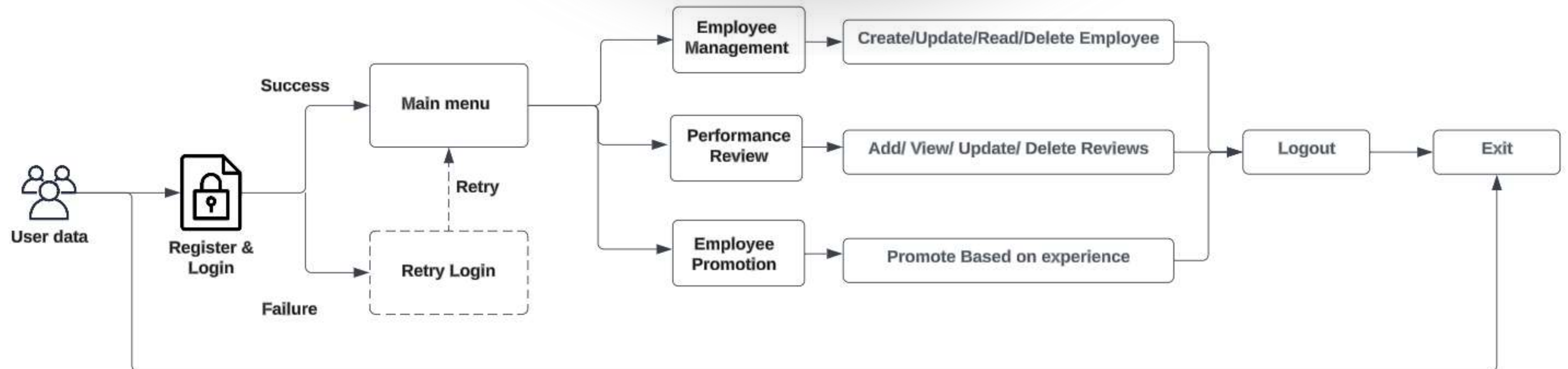
- Console-based user interaction handles inputs and displays outputs.



System Architecture



Software Design





Database Schema

Users Table

Column Name	Data Type	Constraints	Description
id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each user
email	VARCHAR(100)	UNIQUE, NOT NULL	Email address used for login
password	VARCHAR(255)	NOT NULL	Hashed password for authentication
role	ENUM('manager', 'team_lead')	NOT NULL	Role of the user (manager/team_lead)
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Timestamp of user creation



Database Schema

Employees Table

Column Name	Data Type	Constraints	Description
emp_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each employee
user_id	INT	FOREIGN KEY (users.id)	ID of the responsible user
name	VARCHAR(100)	NOT NULL	Full name of the employee
contact_no	VARCHAR(20)		Employee's contact number
address	TEXT		Residential address
dob	DATE		Date of birth
hire_date	DATE		Date hired
position	VARCHAR(100)		Job title/position
salary	DECIMAL(10, 2)		Salary amount
experience	INT		Years of experience



Database Schema

Performance Reviews Table

Column Name	Data Type	Constraints	Description
review_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each review
employee_id	INT	FOREIGN KEY (employees.emp_id)	ID of the reviewed employee
review_date	DATE		Date of the performance review
rating	FLOAT	CHECK (rating >= 1 AND rating <= 5)	Rating given (1 to 5 scale)
feedback	TEXT		Comments on performance

Key Functional Components



Authentication

- Purpose: Securely hash and verify passwords using bcrypt.
- Security: Ensures password confidentiality and integrity through hashing.



Registration

- Takes user input for registration fields.
- Hashes the password.
- Inserts records into users and employees tables.



User login

- Purpose: Authenticates users based on email and password.
- Process:
- Checks input against hashed password in users table.
- Returns user ID on success.



Employee management (CRUD)

- Create: create_employee(user_id)
- Read: list_employees() and read_employee()
- Update: update_employee()
- Delete: delete_employee()



Performance management (CRUD)

- Add Review: add_review()
- View Reviews: view_reviews()
- Update Review: update_review()
- Delete Review: delete_review()



Promote employee

- Function: promote_employee()
- Purpose: Promotes an employee to a new position based on experience.
- Process:
- Lists available positions.
- Updates position and salary based on user selection.

```

22 def register():
23     name = input("Enter Name: ")
24     email = input("Enter Email ID: ")
25     contact = input("Enter Contact No.: ")
26     address = input("Enter Address: ")
27     dob = input("Enter Date of Birth (YYYY-MM-DD): ")
28     hire_date = input("Enter Hire Date (YYYY-MM-DD): ")
29     pw = input("Enter Password: ")
30     confirm_pw = input("Confirm Password: ")
31
32     if pw != confirm_pw:
33         print("Passwords do not match.")
34         return
35
36     hashed_pw = hash_pw(pw)

```

Run: main

```

Enter Email ID: manager@example.com
Enter Contact No.: 1234567890
Enter Address: Paris
Enter Date of Birth (YYYY-MM-DD): 1985-04-02
Enter Hire Date (YYYY-MM-DD): 2020-01-15
Enter Password: 123456
Confirm Password: 123456
Registration successful!

```

Registration

- Registers employees with secure password hashing using bcrypt.
- Inserts user data into users table.

Login

- Logins employees and they can view the main menu
- Main menu inserts data in other 2 tables

```

47 # Login function
48 def login():
49     email = input("Enter Email ID: ")
50     pw = input("Enter Password: ")
51
52     cursor.execute("SELECT id, password FROM users WHERE email = %s", (email,))
53     result = cursor.fetchone()
54
55     if result and check_pw(result[1].encode('utf-8'), pw):
56         print("Login successful! ")
57         return result[0] # Returns user ID (Manager's ID)
58     else:
59         print("Login failed. Check your credentials.")
60         return None
61
62 # Function to list all employees
list_employees()

```

Run: main

```

1. Register
2. Login
3. Exit
Enter your choice: 2
Enter Email ID: hr@example.com
Enter Password: 12345678
Login successful!

```

Add Employee

```
# CRUD: Create new employee
def create_employee(manager_id):
    name = input("Enter Employee Name: ")
    contact = input("Enter Employee Contact No.: ")
    address = input("Enter Employee Address: ")
    dob = input("Enter Employee Date of Birth (YYYY-MM-DD): ")
    hire_date = input("Enter Employee Hire Date (YYYY-MM-DD): ")
    position = input("Enter Employee Position: ")

    try:
        salary = float(input("Enter Employee Salary: "))
    except ValueError:
        print("Please enter a valid salary.")
    return

create_employee()
```

Run: main

Enter Employee Name: Frank Black
Enter Employee Contact No.: 1234567890
Enter Employee Address: Paris
Enter Employee Date of Birth (YYYY-MM-DD): 1991-11-01
Enter Employee Hire Date (YYYY-MM-DD): 2023-01-20
Enter Employee Position: Build Engineer
Enter Employee Salary: 65000
Enter Employee Experience in Years: 2
Employee created successfully!

Read / View

```
# CRUD: Create new employee
def create_employee(manager_id):
    name = input("Enter Employee Name: ")
    contact = input("Enter Employee Contact No.: ")
    address = input("Enter Employee Address: ")
    dob = input("Enter Employee Date of Birth (YYYY-MM-DD): ")
    hire_date = input("Enter Employee Hire Date (YYYY-MM-DD): ")
    position = input("Enter Employee Position: ")

    try:
        salary = float(input("Enter Employee Salary: "))
    except ValueError:
        print("Please enter a valid salary.")
    return

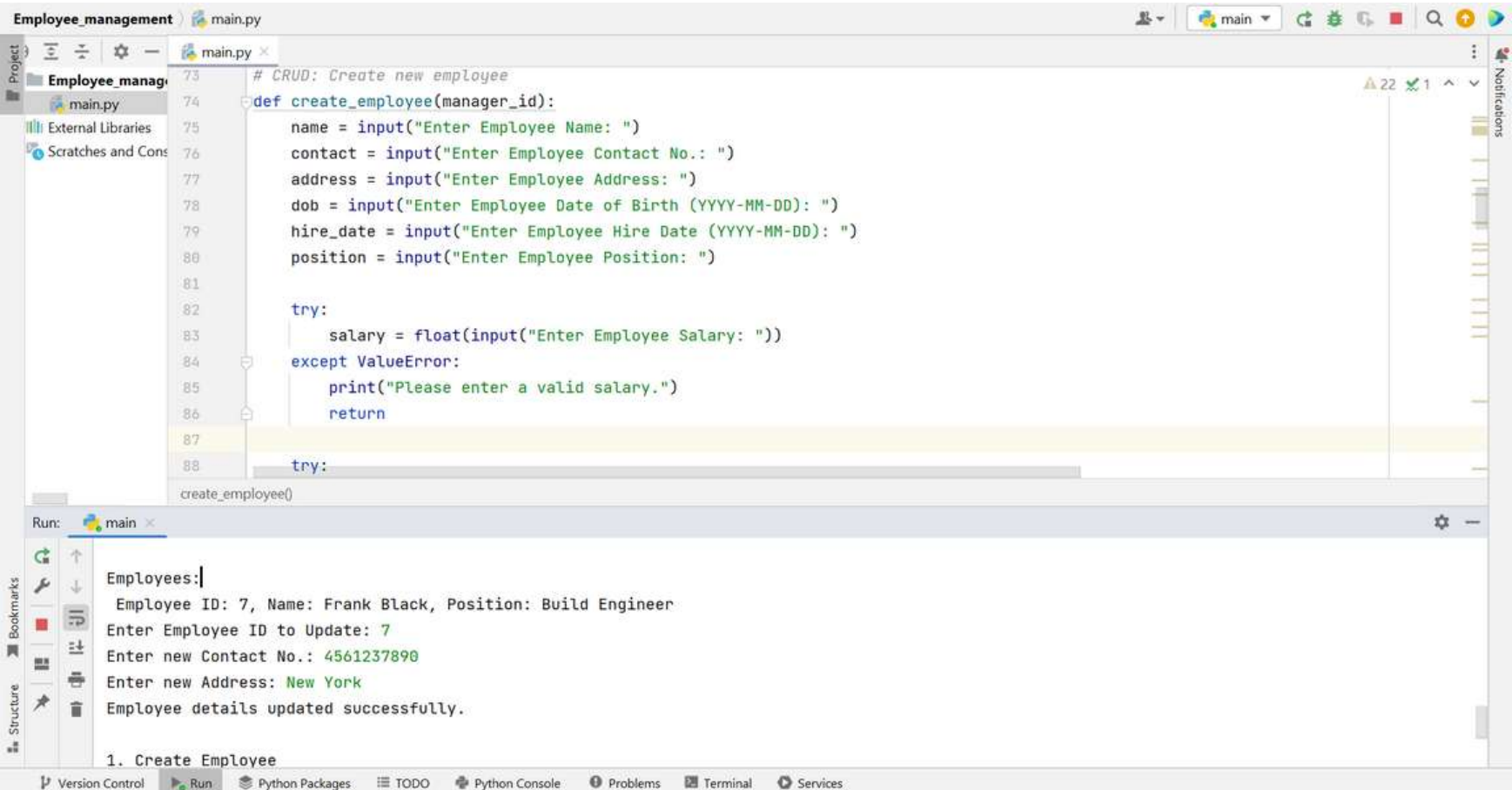
create_employee()
```

Run: main

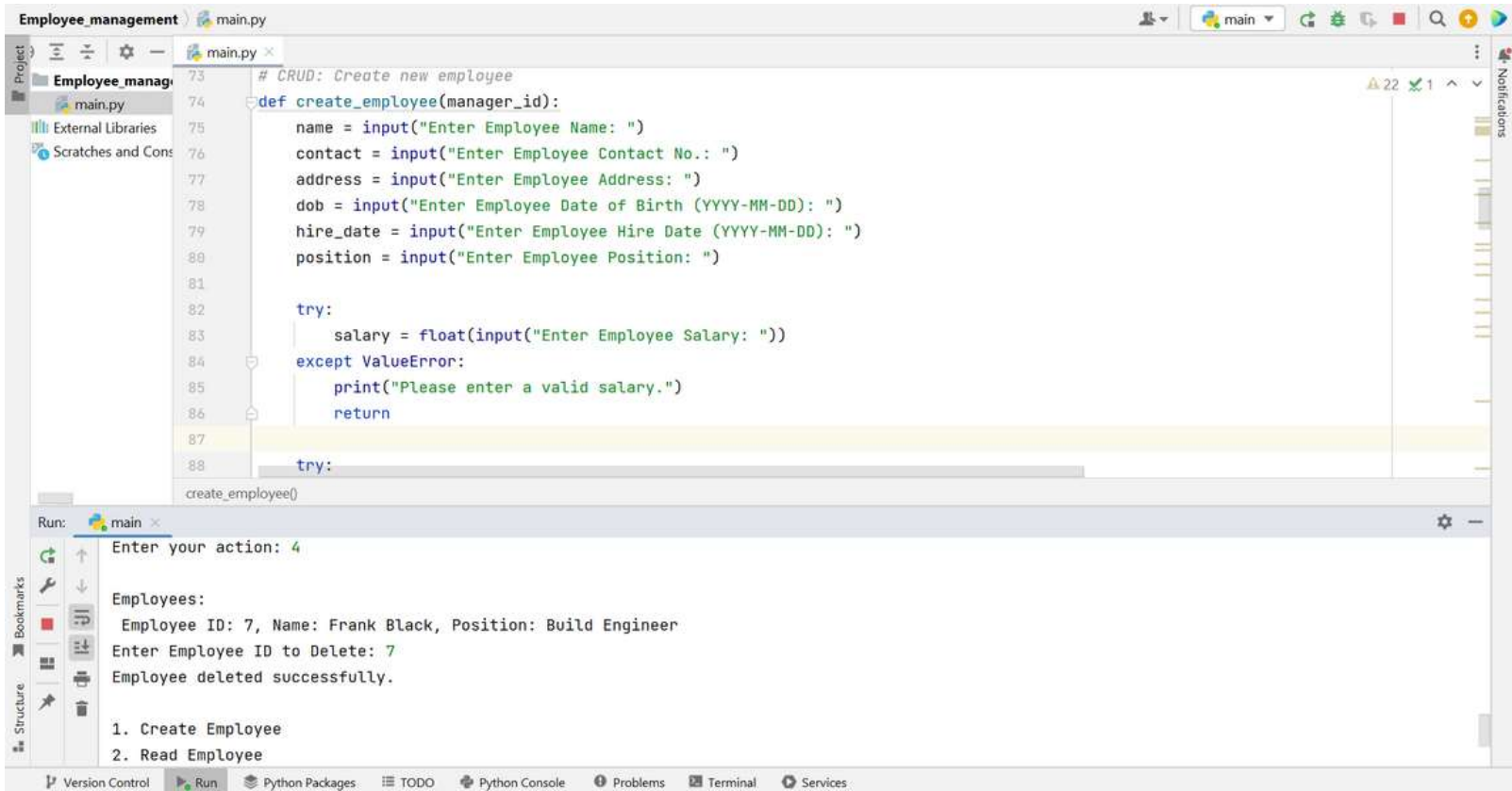
Employees:
Employee ID: 7, Name: Frank Black, Position: Build Engineer
Enter Employee ID to Read: 7
ID: 7, Name: Frank Black, Contact: 1234567890, Address: Paris, DOB: 1991-11-01, Hire Date: 2023-01-20, Position: Build Engineer, Salary: 65000.00, Experience: 2 years

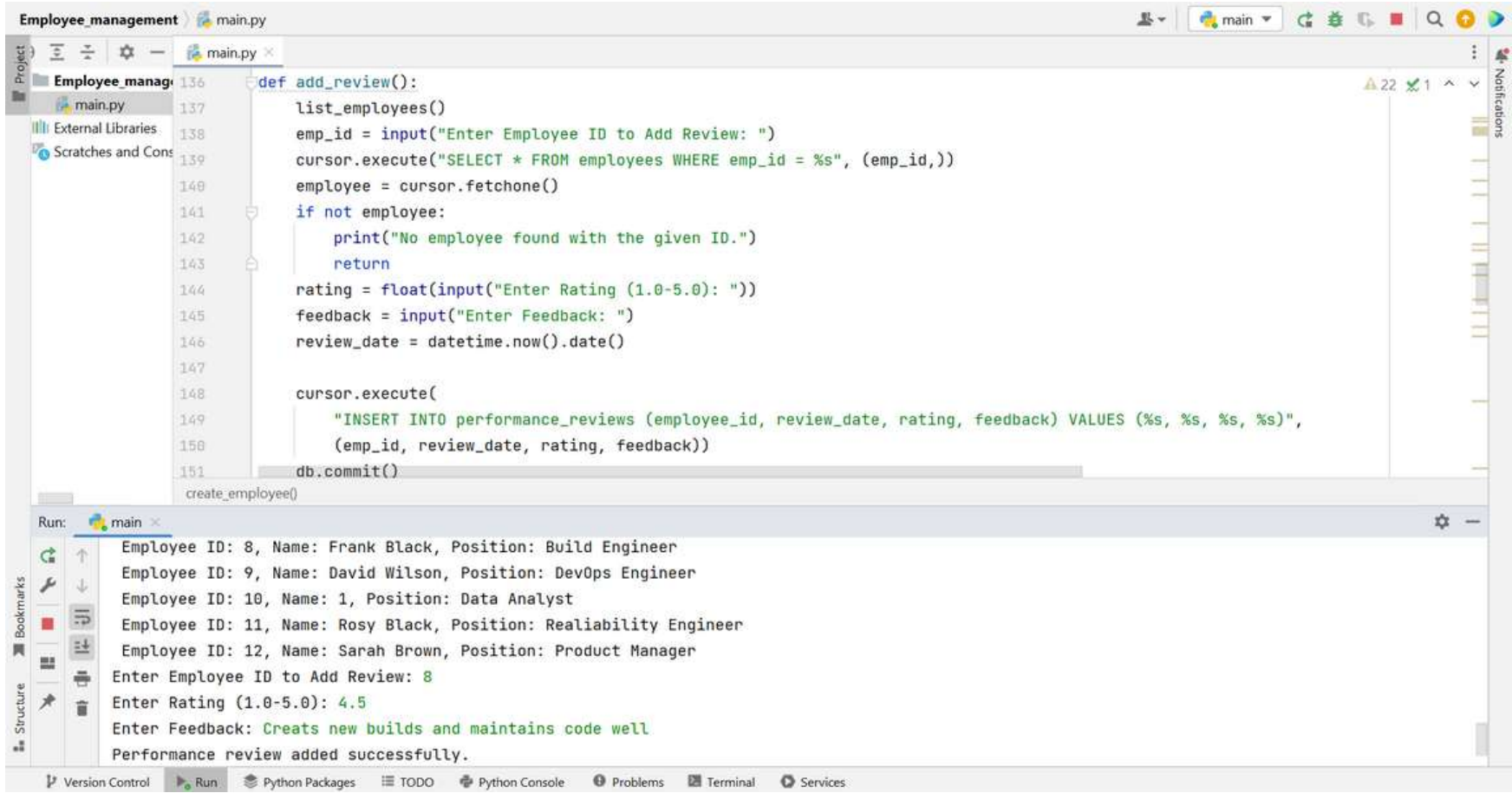
1. Create Employee
2. Read Employee

Update details



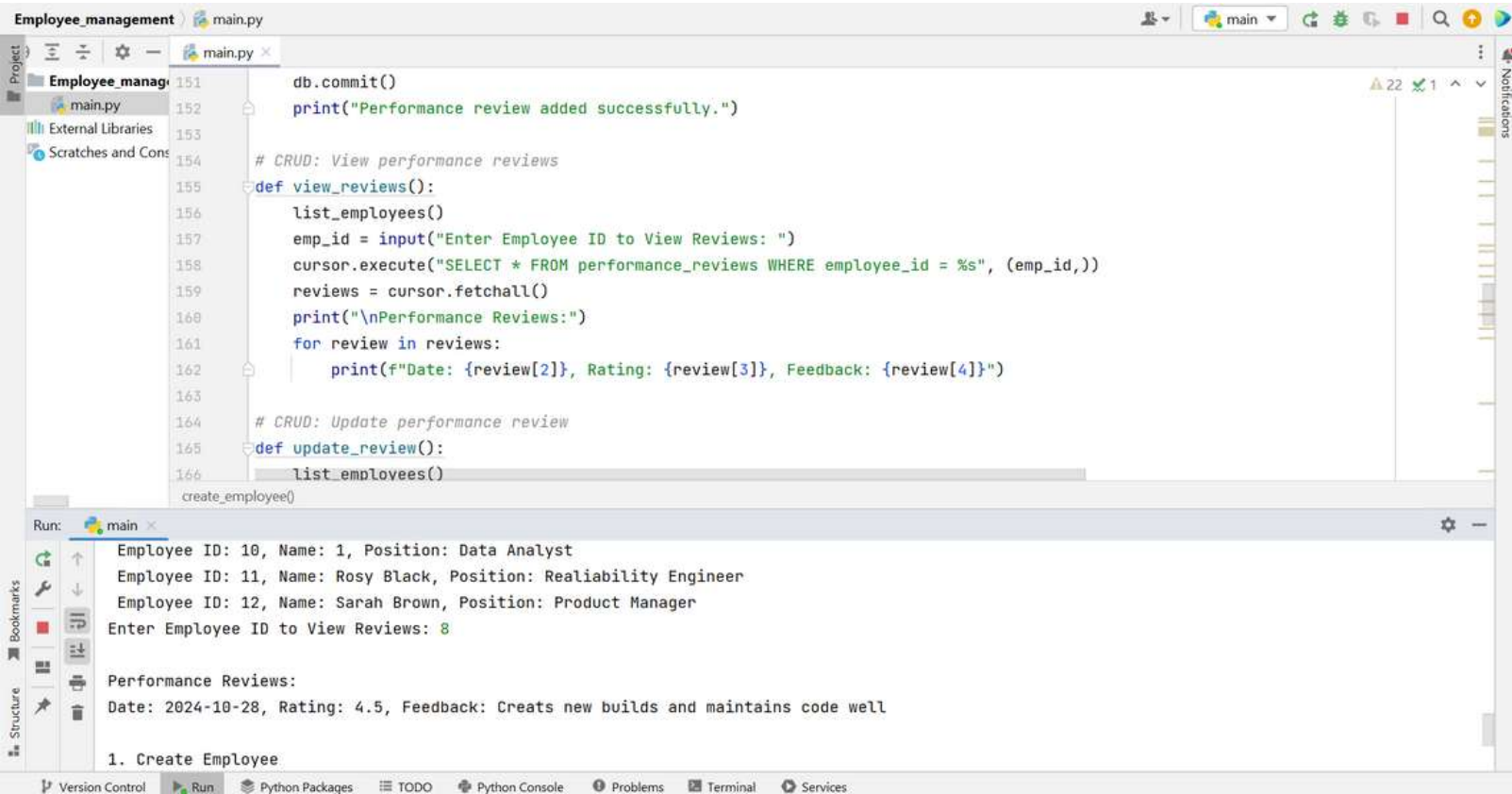
Delete employee





Add Performance review

View Performance Review



Employee_managementmain.py

main.py

```
160 print("\nPerformance Reviews:")
161 for review in reviews:
162     print(f>Date: {review[2]}, Rating: {review[3]}, Feedback: {review[4]}")
163
164 # CRUD: Update performance review
165 def update_review():
166     list_employees()
167     review_id = input("Enter Review ID to Update: ")
168     new_rating = float(input("Enter new Rating (1.0-5.0): "))
169     new_feedback = input("Enter new Feedback: ")
170     cursor.execute("UPDATE performance_reviews SET rating = %s, feedback = %s WHERE review_id = %s",
171                   (new_rating, new_feedback, review_id))
172     db.commit()
173     print("Review updated successfully.")
174
175 # CRUD: Delete performance review
create_employee()
```

Run: main

Employee ID: 8, Name: Frank Black, Position: Build Engineer
Employee ID: 9, Name: David Wilson, Position: DevOps Engineer
Employee ID: 10, Name: 1, Position: Data Analyst
Employee ID: 11, Name: Rosy Black, Position: Realiability Engineer
Employee ID: 12, Name: Sarah Brown, Position: Product Manager
Enter Review ID to Update: 8
Enter new Rating (1.0-5.0): 3.9
Enter new Feedback: manages code well but needs more improvement.
Review updated successfully.

Update Performance review

Delete Performance Review

Employee_managementmain.py

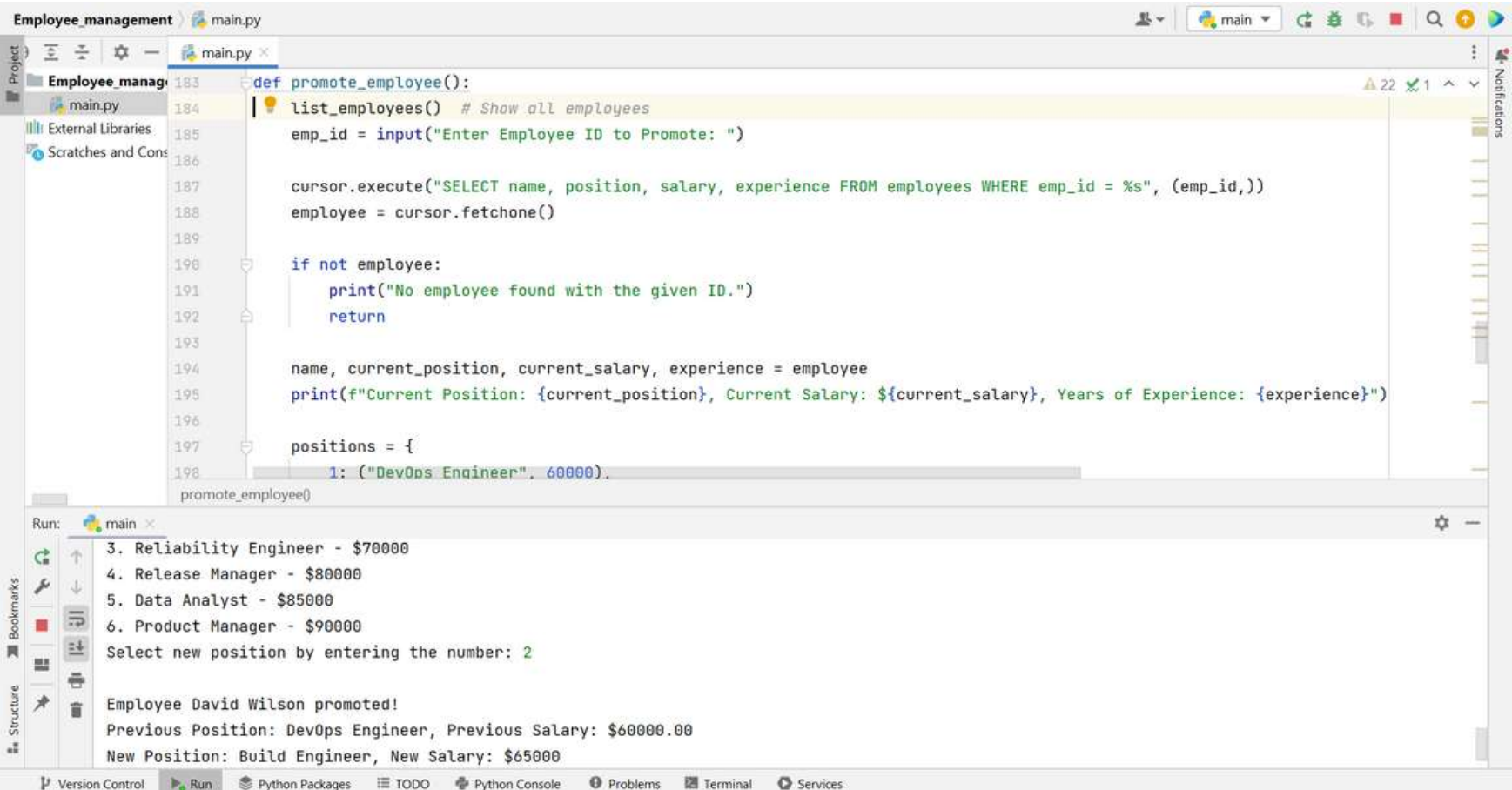
main.py

```
172 db.commit()
173 print("Review updated successfully.")
174
175 # CRUD: Delete performance review
176 def delete_review():
177     list_employees()
178     review_id = input("Enter Review ID to Delete: ")
179     cursor.execute("DELETE FROM performance_reviews WHERE review_id = %s", (review_id,))
180     db.commit()
181     print("Review deleted successfully.")
182
183 def promote_employee():
184     list_employees() # Show all employees
185     emp_id = input("Enter Employee ID to Promote: ")
186
187     cursor.execute("SELECT name, position, salary, experience FROM employees WHERE emp_id = %s". (emp_id,))
create_employee()
```

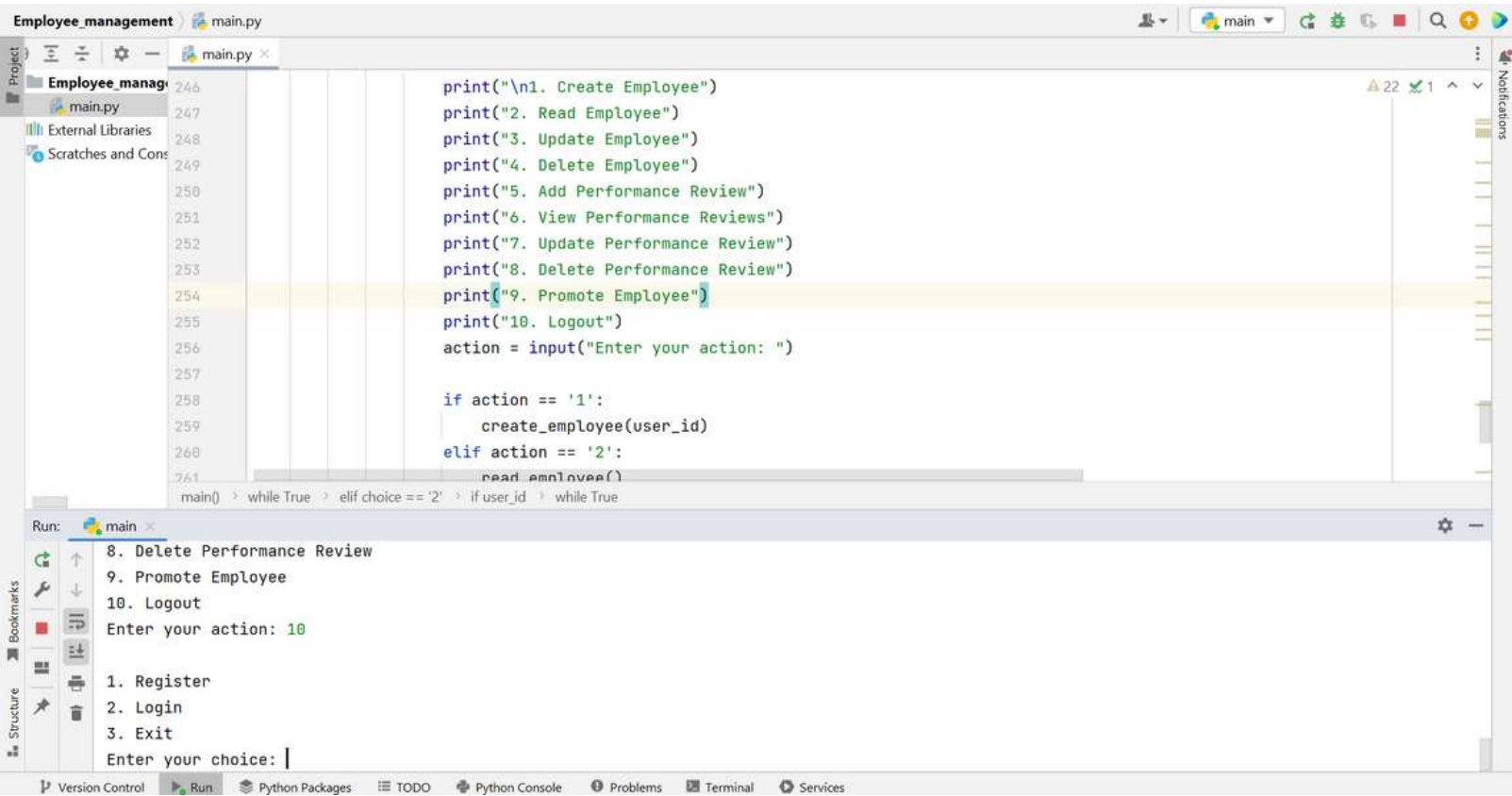
Run: main

Employees:
Employee ID: 8, Name: Frank Black, Position: Build Engineer
Employee ID: 9, Name: David Wilson, Position: DevOps Engineer
Employee ID: 10, Name: 1, Position: Data Analyst
Employee ID: 11, Name: Rosy Black, Position: Realiability Engineer
Employee ID: 12, Name: Sarah Brown, Position: Product Manager
Enter Review ID to Delete: 8
Review deleted successfully.

Promote Employee



Logout



Conclusion

The Employee Performance Management System (EPMS) is designed to streamline employee and performance management in a secure and efficient manner. The system allows for CRUD operations on employees and reviews, secure password management, and promotion handling, while following a layered architecture. Future enhancements will focus on improving usability, security, and scalability.





Thank you

