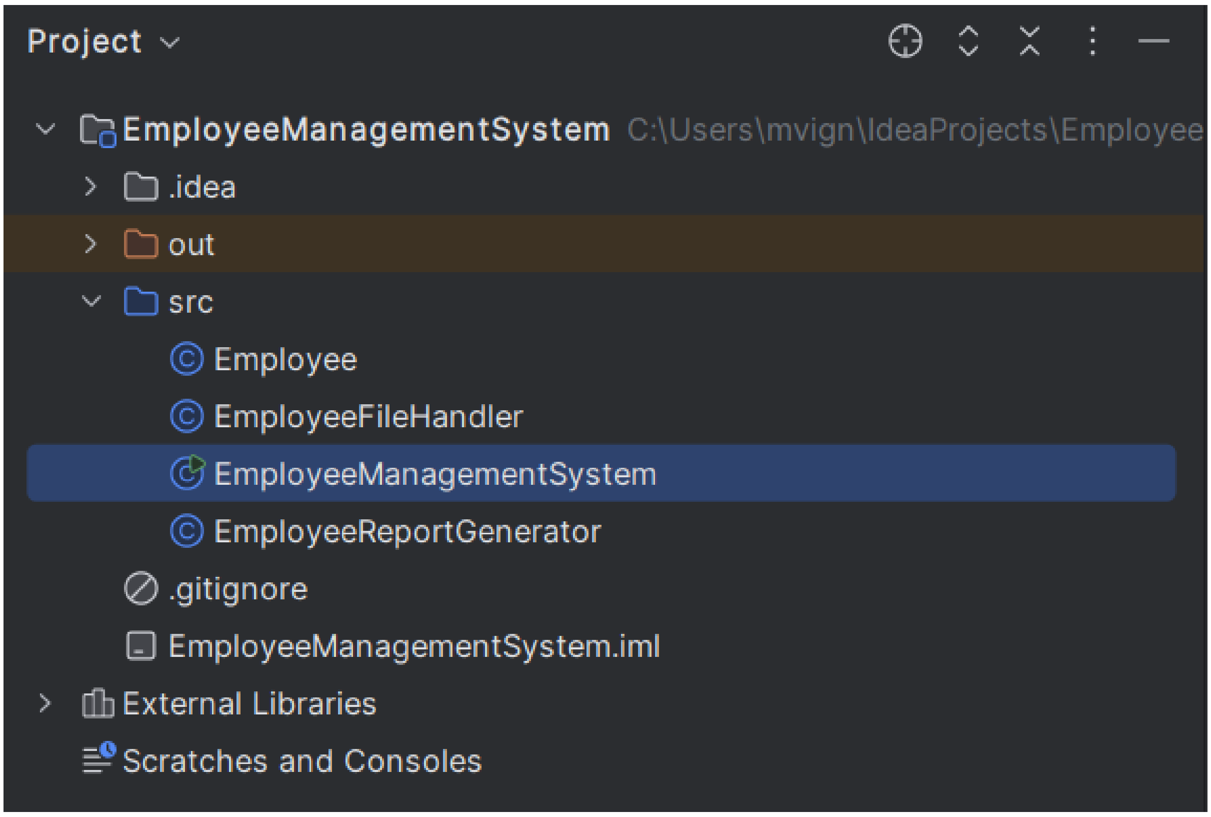
# Employee Management System

Code structure:



EmployeeManagementSystem/ │ ├── Employee.java ├── EmployeeManagementSystem.java ├── EmployeeFileHandler.java ├── EmployeeReportGenerator.java ├── employees.dat ├── README.md └── documentation.md

Code:

## Employee.java

```
import java.io.Serializable;

import java.time.LocalDate;


public class Employee implements Serializable {

 private static final long serialVersionUID = 1L;
private String id;

private String name;

private String department;

private String position;

private double salary;

private LocalDate joinDate;


public Employee(String id, String name, String department, String position, double salary,
LocalDate joinDate) {

   this.id = id;

   this.name = name;

   this.department = department;

   this.position = position;

   this.salary = salary;

   this.joinDate = joinDate;

}


public String getId() { return id; }

public String getName() { return name; }

public String getDepartment() { return department; }

public String getPosition() { return position; }

public double getSalary() { return salary; }

public LocalDate getJoinDate() { return joinDate; }


public void setName(String name) { this.name = name; }

public void setDepartment(String department) { this.department = department; }

public void setPosition(String position) { this.position = position; }

public void setSalary(double salary) { this.salary = salary; }


}
```

## EmployeeFileHandler.java

```
import java.io.*; import java.util.List;


public class EmployeeFileHandler {

private static final String FILE_NAME = "employees.dat";


public static void saveToFile(List<Employee> employees) {

  try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {

    oos.writeObject(employees);

    System.out.println("Employee data saved to file.");

  } catch (IOException e) {

    System.out.println("Error saving file.");

  }

}


@SuppressWarnings("unchecked")

public static List<Employee> loadFromFile() {

  try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {

    return (List<Employee>) ois.readObject();

  } catch (Exception e) {

    System.out.println("No existing data found.");

    return null;

  }

}


}
```

## EmployeeReportGenerator.java

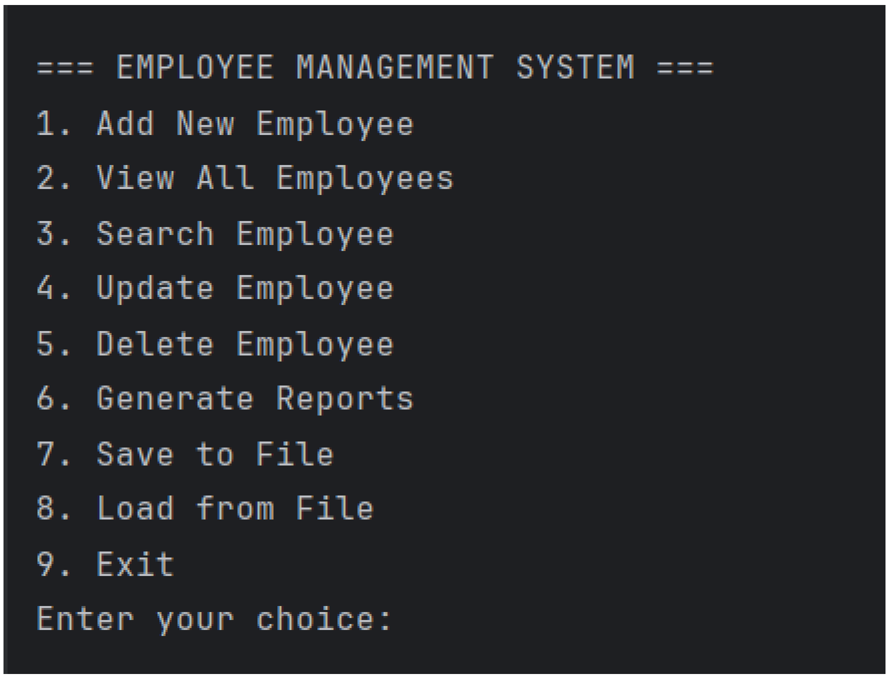
```
import java.util.*;

public class EmployeeReportGenerator { public static void
generateDepartmentReport(List<Employee> employees) { Map<String, List<Employee>> map =
new HashMap<>();

for (Employee e : employees) {
    map.computeIfAbsent(e.getDepartment(), k -> new ArrayList<>()).add(e);
  }

  for (String dept : map.keySet()) {
    List<Employee> list = map.get(dept);
    double avg = list.stream().mapToDouble(Employee::getSalary).average().orElse(0);
    System.out.println(dept + ": " + list.size() + " employees, Average: ₹" + String.format("%.2f",
avg));
  }
}

}
```

## EmployeeManagementSystem.java

```
import java.time.LocalDate; import java.util.*;

public class EmployeeManagementSystem { private static List<Employee> employees = new
ArrayList<>(); private static Scanner sc = new Scanner(System.in);

public static void main(String[] args) {
  List<Employee> loaded = EmployeeFileHandler.loadFromFile();
  if (loaded != null) employees = loaded;


  while (true) {
    System.out.println("\n=== EMPLOYEE MANAGEMENT SYSTEM ===");
    System.out.println("1. Add New Employee");
    System.out.println("2. View All Employees");
    System.out.println("3. Search Employee");
    System.out.println("4. Update Employee");
    System.out.println("5. Delete Employee");
    System.out.println("6. Generate Reports");
    System.out.println("7. Save to File");
    System.out.println("8. Load from File");
    System.out.println("9. Exit");
    System.out.print("Enter your choice: ");


    int choice = sc.nextInt();
    sc.nextLine();


    switch (choice) {
      case 1 -> addEmployee();
      case 2 -> viewEmployees();
      case 3 -> searchEmployee();
      case 4 -> updateEmployee();
      case 5 -> deleteEmployee();
      case 6 -> EmployeeReportGenerator.generateDepartmentReport(employees);
      case 7 -> EmployeeFileHandler.saveToFile(employees);
      case 8 -> employees =
Optional.ofNullable(EmployeeFileHandler.loadFromFile()).orElse(employees);
      case 9 -> System.exit(0);
      default -> System.out.println("Invalid choice");
    }
```

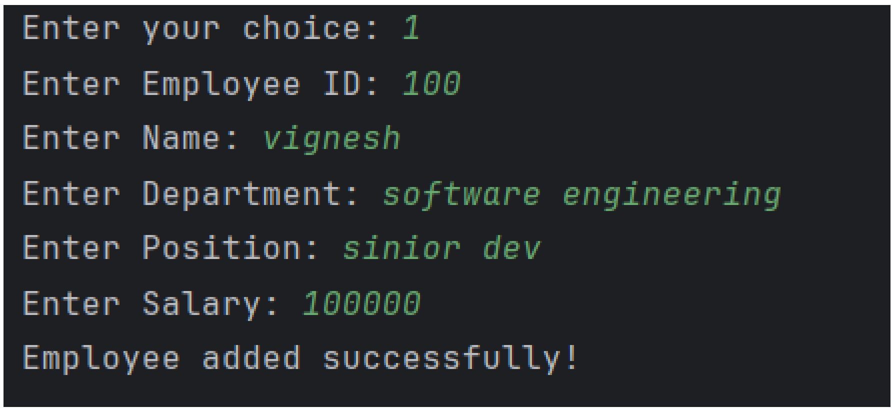
```java
    }
}

private static void addEmployee() {
    System.out.print("Enter Employee ID: ");
    String id = sc.nextLine();
    System.out.print("Enter Name: ");
    String name = sc.nextLine();
    System.out.print("Enter Department: ");
    String dept = sc.nextLine();
    System.out.print("Enter Position: ");
    String pos = sc.nextLine();
    System.out.print("Enter Salary: ");
    double sal = sc.nextDouble();
    sc.nextLine();

    Employee emp = new Employee(id, name, dept, pos, sal, LocalDate.now());
    employees.add(emp);
    System.out.println("Employee added successfully!");
}

private static void viewEmployees() {
    System.out.println("\nID\tName\tDepartment\tPosition\tSalary\tJoin Date");
    for (Employee e : employees) {
        System.out.println(e.getId() + "\t" + e.getName() + "\t" + e.getDepartment() + "\t" +
                e.getPosition() + "\t₹" + e.getSalary() + "\t" + e.getJoinDate());
    }
}

private static void searchEmployee() {
    System.out.print("Enter Employee ID: ");
    String id = sc.nextLine();
    for (Employee e : employees) {
        if (e.getId().equalsIgnoreCase(id)) {
            System.out.println("Found: " + e.getName() + ", " + e.getDepartment());
            return;
        }
    }
    System.out.println("Employee not found");
}

private static void updateEmployee() {
    System.out.print("Enter Employee ID to update: ");
    String id = sc.nextLine();
    for (Employee e : employees) {
        if (e.getId().equalsIgnoreCase(id)) {
            System.out.print("New Salary: ");
            e.setSalary(sc.nextDouble());
            sc.nextLine();
            System.out.println("Employee updated");
            return;
        }
    }
    System.out.println("Employee not found");
}

private static void deleteEmployee() {
    System.out.print("Enter Employee ID to delete: ");
    String id = sc.nextLine();
    employees.removeIf(e -> e.getId().equalsIgnoreCase(id));
    System.out.println("Employee deleted if existed");
}}
```

Outputs:



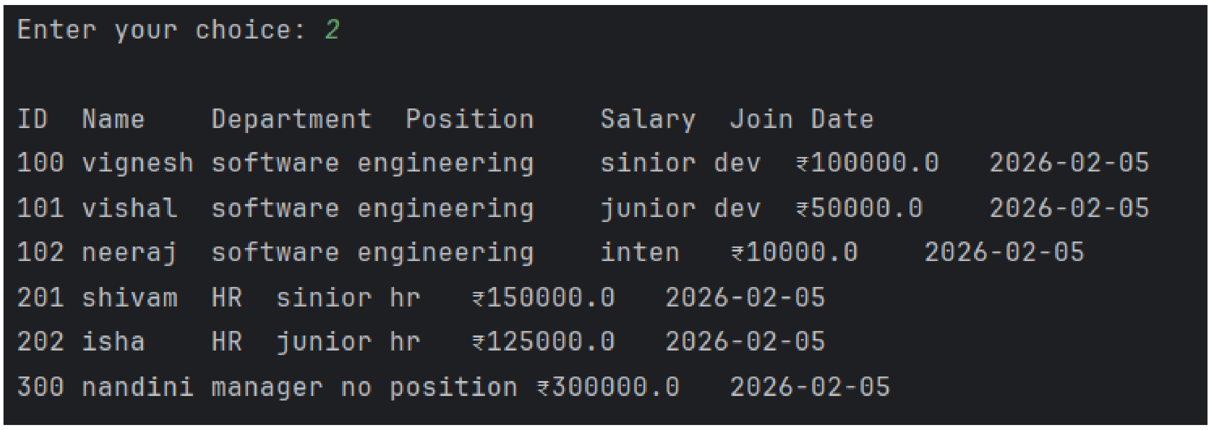
```
=== EMPLOYEE MANAGEMENT SYSTEM ===
1. Add New Employee
2. View All Employees
3. Search Employee
4. Update Employee
5. Delete Employee
6. Generate Reports
7. Save to File
8. Load from File
9. Exit
Enter your choice:
```

Choice1

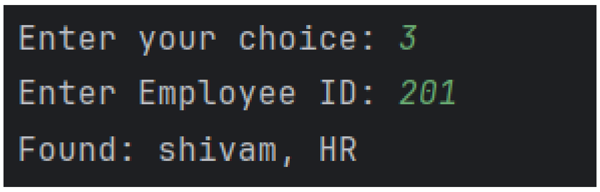


```
Enter your choice: 1
Enter Employee ID: 100
Enter Name: vignesh
Enter Department: software engineering
Enter Position: sinior dev
Enter Salary: 100000
Employee added successfully!
```

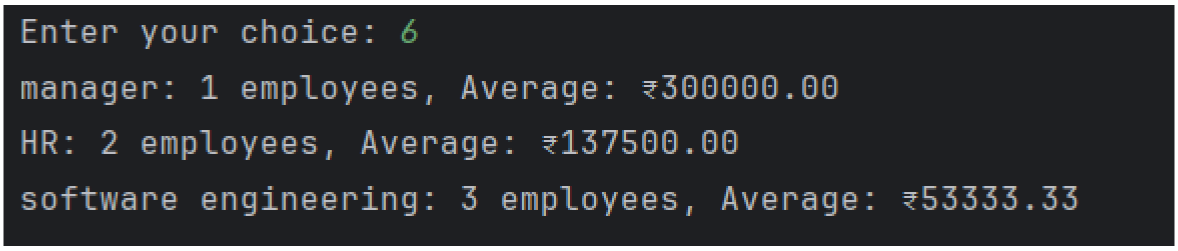
Choice 2:



```
Enter your choice: 2

ID  Name    Department  Position    Salary  Join Date
100 vignesh software engineering    sinior dev  ₹100000.0   2026-02-05
101 vishal  software engineering    junior dev  ₹50000.0    2026-02-05
102 neeraj  software engineering    inten   ₹10000.0    2026-02-05
201 shivam  HR  sinior hr   ₹150000.0   2026-02-05
202 isha    HR  junior hr   ₹125000.0   2026-02-05
300 nandini manager no position ₹300000.0   2026-02-05
```



```
Enter your choice: 3
Enter Employee ID: 201
Found: shivam, HR
```



```
Enter your choice: 6
manager: 1 employees, Average: ₹300000.00
HR: 2 employees, Average: ₹137500.00
software engineering: 3 employees, Average: ₹53333.33
```