Add new employees: The user can add details like employee ID, name, department, and salary.

a) Update employee details: The user can update the name, department, or salary of an existing employee based on their employee ID.
b) Delete an employee: The user can delete an employee from the system based on their employee ID.
c) Display all employees: The user can view a list of all employees and their details.
d) Search for an employee: The user can search for an employee by their employee ID and view their details.
e) Requirements:
    i) Use Object-Oriented Programming (OOP) principles and create an Employee class with appropriate attributes and methods.
    ii) Use appropriate data structures (e.g., HashMap) to store the employee data.
    iii) Implement exception handling to handle possible errors (e.g., invalid employee ID, input validation).
    iv) Provide a user-friendly console interface for the user to interact with the Employee Management System.

Implementation: As we know Employee management system that user use to manage employees of any organization or any company same as given task I created Employee Management System to manage employees, this system fulfilled the all prerequisites as given in the task, also I implemented all the concepts of OOPs, Interface and collections to manipulate employees.

This Employee class consist appropriate attributes that shows basic but need full parameters for employee and appropriates to instantiate, instance data members.

```
package com.Assignments.EmployeeManagementSystem;

public class Employee {

	private String id;
	private String name;
	private String department;
	private double salary;

	public Employee() {
		super();
	}

	public Employee(String id, String name, String department, double sal-
ary) {
		super();
		this.id = id;
		this.name = name;
		this.department = department;
		this.salary = salary;
```

```
	}

	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

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

	public String getDepartment() {
		return department;
	}

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

	public double getSalary() {
		return salary;
	}

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

	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", department="
+ department + ", salary=" + salary + "]";
	}
}
```

This interface shows the data integrity and confidentiality using data abstraction, this interface somewhere act as design pattern DAO (Data Access Object).

```
package com.Assignments.EmployeeManagementSystem;


public interface EmployeeDAO {

	public abstract void addEmployee(String id, String name, String depart-
ment, double salary);

	public abstract void updateEmployee(String id, String name, String de-
partment, double salary);

	public abstract void deleteEmployee(String id);

	public abstract void searchEmployee(String id);

	public abstract void displayAllEmployees();
}
```

As we having Interface that are responsible for data abstraction, so this is the **EmployeeDAOImpl** class act as implement class of **EmployeeDAO** interface to implement unimplemented methods with some owned feature, each method comes with different behavior that very need full for our core concept that is employee management system.

As we know for any Employee, Employee ID it is very important key point to know information of any employee, to implement that concept called MAP interface to store value in the form of key and pair, here key is Employee ID and others information.

```
package com.Assignments.EmployeeManagementSystem;

import java.util.HashMap;
import java.util.Map;

public class EmployeeDAOImpl implements EmployeeDAO {

	private Map<String, Employee> employees;

	public EmployeeDAOImpl() {
		employees = new HashMap<>();
	}
```

```
	@Override
	public void addEmployee(String id, String name, String department, dou-
ble salary) {
		if (employees.containsKey(id)) {
			System.out.println("Error: Employee with ID " + id + " al-
ready exists.");
			return;
		}
		Employee employee = new Employee(id, name, department, salary);
		employees.put(id, employee);
		System.out.println("Employee with ID " + id + " added success-
fully.");
	}

	@Override
	public void updateEmployee(String id, String name, String department,
double salary) {
		Employee employee = employees.get(id);
		if (employee == null) {
			System.out.println("Error: Employee with ID " + id + " does
not exist.");
			return;
		}
		if (name != null && !name.isEmpty()) {
			employee.setName(name);
		}
		if (department != null && !department.isEmpty()) {
			employee.setDepartment(department);
		}
		if (salary > 0) {
			employee.setSalary(salary);
		}
		System.out.println("Employee with ID " + id + " updated success-
fully.");
	}

	@Override
	public void deleteEmployee(String id) {
		if (!employees.containsKey(id)) {
			System.out.println("Error: Employee with ID " + id + " does
not exist.");
			return;
		}
		employees.remove(id);
		System.out.println("Employee with ID " + id + " deleted success-
fully.");
	}

	@Override
	public void searchEmployee(String id) {
```

```
            Employee employee = employees.get(id);
            if (employee == null) {
                  System.out.println("Error: Employee with ID " + id + " does
not exist.");
                  return;
            }
            System.out.println(employee.toString());
      }

      @Override
      public void displayAllEmployees() {
            if (employees.isEmpty()) {
                  System.out.println("No employees to display.");
                  return;
            }
            for (Employee employee : employees.values()) {
                  System.out.println(employee.toString());
            }
      }
}
```

This is the main class that provides user-friendly console to manage visible data and manipulate data any end user. It is provides after started unstoppable menus until user don't shut the system also it provides all necessary menus for any HR or end-user to manage Employee Management System.

```
package com.Assignments.EmployeeManagementSystem;

import java.util.Scanner;

class EmployeeManagementSystem {

      public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);
            EmployeeDAOImpl employeeImpl= new EmployeeDAOImpl();

            while (true) {
                  System.out.println("\nEmployee Management System");
                  System.out.println("1. Add Employee");
                  System.out.println("2. Update Employee");
                  System.out.println("3. Delete Employee");
                  System.out.println("4. Display All Employees");
                  System.out.println("5. Search Employee");
                  System.out.println("6. Exit");

                  System.out.print("Enter your choice: ");
                  String choice = scanner.nextLine();
```

```
                switch (choice) {
                case "1":
                        System.out.print("Enter employee ID: ");
                        String id = scanner.nextLine();
                        System.out.print("Enter name: ");
                        String name = scanner.nextLine();
                        System.out.print("Enter department: ");
                        String department = scanner.nextLine();
                        System.out.print("Enter salary: ");
                        double salary = scanner.nextDouble();
                        scanner.nextLine(); // Consume newline
                        employeeImpl.addEmployee(id, name, department, sal-
ary);
                        break;
                case "2":
                        System.out.print("Enter employee ID: ");
                        id = scanner.nextLine();
                        System.out.print("Enter new name (leave blank if no
change): ");
                        name = scanner.nextLine();
                        System.out.print("Enter new department (leave blank
if no change): ");
                        department = scanner.nextLine();
                        System.out.print("Enter new salary (leave blank if no
change): ");
                        String salaryInput = scanner.nextLine();
                        salary = salaryInput.isEmpty() ? -1 : Dou-
ble.parseDouble(salaryInput);
                        employeeImpl.updateEmployee(id, name, department,
salary);
                        break;
                case "3":
                        System.out.print("Enter employee ID: ");
                        id = scanner.nextLine();
                        employeeImpl.deleteEmployee(id);
                        break;
                case "4":
                        employeeImpl.displayAllEmployees();
                        break;
                case "5":
                        System.out.print("Enter employee ID: ");
                        id = scanner.nextLine();
                        employeeImpl.searchEmployee(id);
                        break;
                case "6":
                        System.out.println("Exiting...");
                        return;

                default:
```

```
                        System.out.println("Invalid choice. Please try
again.");
                    }
                }
            }
}
```

OUTPUT: As given in the task all requisites we can execute, shown in further output.

```
Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Search Employee
6. Exit
Enter your choice: 1
Enter employee ID: SDE1001
Enter name: Mark Juckerberg
Enter department: Facebook
Enter salary: 20000
Employee with ID SDE1001 added successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Search Employee
6. Exit
Enter your choice: 1
Enter employee ID: SDE2001
Enter name: Jeff Bezos
Enter department: Amazon
Enter salary: 25000
Employee with ID SDE2001 added successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Search Employee
6. Exit
Enter your choice: 1
Enter employee ID: SDE1002
Enter name: Saurabh Yadav Hedau
Enter department: Facebook
```

```
Enter salary: 15000
Employee with ID SDE1002 added successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Search Employee
6. Exit
Enter your choice: 4
Employee [id=SDE2001, name=Jeff Bezos, department=Amazon, salary=25000.0]
Employee [id=SDE1001, name=Mark Juckerberg, department=Facebook, sal-
ary=20000.0]
Employee [id=SDE1002, name=Saurabh Yadav Hedau, department=Facebook, sal-
ary=15000.0]

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Search Employee
6. Exit
Enter your choice: 2
Enter employee ID: SDE1002
Enter new name (leave blank if no change): SAURABH HEDAU
Enter new department (leave blank if no change):
Enter new salary (leave blank if no change): 17500
Employee with ID SDE1002 updated successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Search Employee
6. Exit
Enter your choice: 4
Employee [id=SDE2001, name=Jeff Bezos, department=Amazon, salary=25000.0]
Employee [id=SDE1001, name=Mark Juckerberg, department=Facebook, sal-
ary=20000.0]
Employee [id=SDE1002, name=SAURABH HEDAU, department=Facebook, sal-
ary=17500.0]

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
```

```
5. Search Employee
6. Exit
Enter your choice: 3
Enter employee ID: SDE1002
Employee with ID SDE1002 deleted successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Search Employee
6. Exit
Enter your choice: 4
Employee [id=SDE2001, name=Jeff Bezos, department=Amazon, salary=25000.0]
Employee [id=SDE1001, name=Mark Juckerberg, department=Facebook, sal-
ary=20000.0]

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Search Employee
6. Exit
Enter your choice: 5
Enter employee ID: SDE1002
Error: Employee with ID SDE1002 does not exist.

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Search Employee
6. Exit
Enter your choice: 5
Enter employee ID: SDE2001
Employee [id=SDE2001, name=Jeff Bezos, department=Amazon, salary=25000.0]

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Search Employee
6. Exit
Enter your choice: 5
Enter employee ID: SDE1001
```

```
Employee [id=SDE1001, name=Mark Juckerberg, department=Facebook, sal-
ary=20000.0]


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