Employee Management System (EMS) - Project Documentation

📘 Project Title:

Employee Management System (Console-Based Java Application)

👥 Team:

Developed by V18 Team

1. Ujjwal kumar (Leader)
2. Vikash kumar
3. Ritik kumar

📝 Introduction

The Employee Management System (EMS) is a console-based Java application designed to provide a simple yet effective solution for managing employee records. It is developed with the primary goal of helping small organizations or teams maintain their employee data efficiently without requiring complex database setups.

This system allows users to perform essential operations such as adding new employee details, viewing existing records, updating specific employee information, and removing employee data, all through a user-friendly command-line interface. Each employee's data is stored as a uniquely named text file, enabling persistent storage while keeping the system lightweight and easy to maintain.

The EMS project emphasizes robustness and data integrity by incorporating comprehensive input validation and error handling techniques. Users are guided with clear prompts and messages to prevent invalid data entry and to handle unexpected errors gracefully. The modular design divides functionality across dedicated classes, promoting code readability, maintainability, and seamless integration of components.

Key innovative features include partial field updates, structured and readable data storage format, and a well-organized console UI that enhances usability. This project serves both as a practical tool for employee record management and as an educational platform for demonstrating core programming concepts such as file handling, input validation, modular programming, and error management in Java.

Overall, the Employee Management System is a reliable, extensible, and easy-to-use solution that meets the fundamental needs of employee data management with a clean design and efficient performance.

---

1. 🛠 Core Feature Implementation

The EMS application includes the following core features:

- **Add Employee**: Captures full employee details and stores them in a unique text file.
- **View Employee**: Retrieves and displays employee details based on the ID.
- **Remove Employee**: Deletes the record file of the specified employee.
- **Update Employee**: Allows users to update any specific detail of an employee.
- **Exit**: Cleanly exits the application.

Each feature is implemented in its dedicated class for clarity and separation of concerns.

---

2. ❗ Error Handling and Robustness

The system gracefully handles various types of errors:

- **File Not Found**: Displays a message if the requested employee record does not exist.
- **Duplicate Entry**: Prevents overwriting an existing employee file.
- **Invalid Input**: Shows specific error messages and blocks faulty data from being processed.
- **I/O Exceptions**: All file operations are wrapped in try-catch blocks to handle exceptions.

---

3. 🔄 Integration of Components

All modules interact through the main class, ensuring seamless integration:

- MainMenu displays UI options.
- EmployeeAdd, EmployeeShow, EmployeeRemove, EmployeeUpdate are called based on user input.

- Shared utilities like Scanner and ValidationUtils reduce code duplication.
- Consistent file naming (data/file<ID>.txt) ensures smooth file operations.

---

4. ⚙ Event Handling and Processing

Handled via a while(true) loop in main() using a switch-case structure:

- User input is processed as events triggering corresponding actions.
- Incorrect choices are handled with error messages.
- After every operation, the user is prompted to continue.

---

5. ✔ Data Validation

Validation is centralized in a dedicated utility class:

- isNonEmpty(String) — checks for empty input.
- isValidEmail(String) — regex validation for email.
- isValidPhone(String) — ensures 10-digit numeric contact.
- isValidSalary(String) — allows valid numeric values with up to two decimal places.

All fields are validated before storing or updating employee data.

---

6. ▯ Code Quality and Innovative Features

Code Quality:

- Modular classes with clear responsibilities.
- Reusable utility methods (ValidationUtils).
- Consistent naming conventions.
- Well-commented code for maintainability.

Innovative Features:

- 🗀 Lightweight file-based storage.
- ▯ Readable formatting of stored employee data.
- ⚠ Field-specific validation and error messages.

- 🔁 Partial field updates.
- 🎨 Styled console UI with structured output.

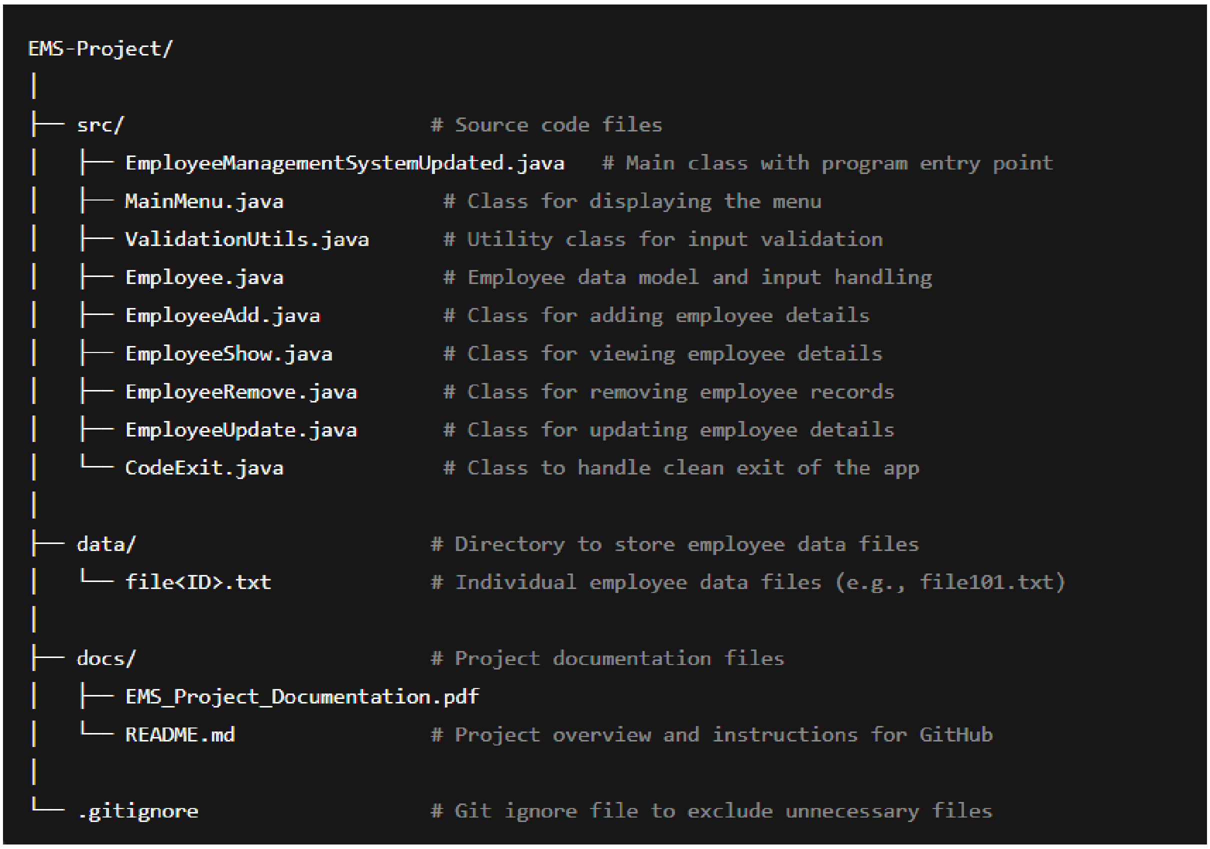
Optional Features for Future:

- Search by name/contact.
- View all employees in tabular format.
- Login authentication system.
- Export data to CSV.

📂 Directory Structure:

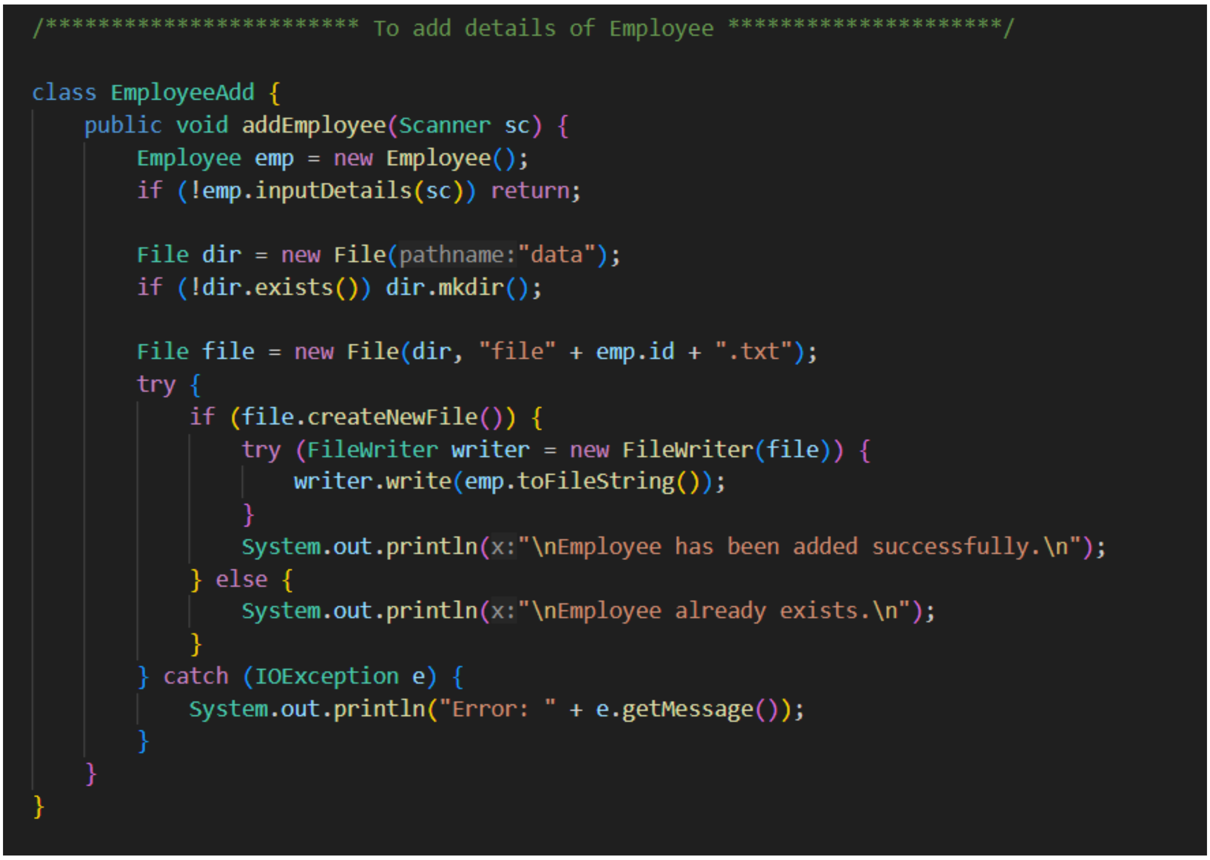
```
EMS/
├── data/
│   └── file<ID>.txt
├── EmployeeManagementSystemUpdated.java
└── (Other class files if separated)
```

📂 Project Structure and Folder Hierarchy

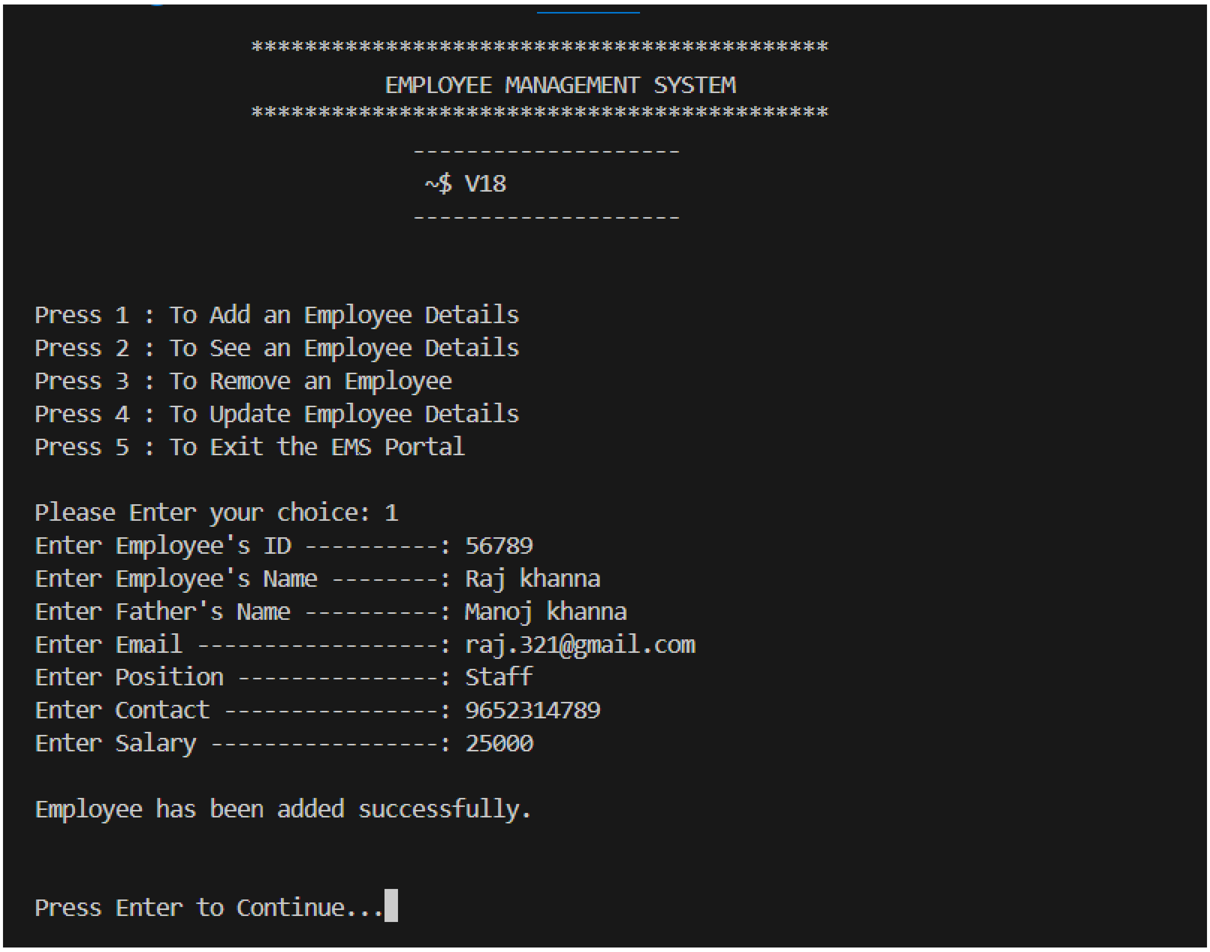


```
EMS-Project/
│
├── src/                          # Source code files
│   ├── EmployeeManagementSystemUpdated.java    # Main class with program entry point
│   ├── MainMenu.java              # Class for displaying the menu
│   ├── ValidationUtils.java       # Utility class for input validation
│   ├── Employee.java              # Employee data model and input handling
│   ├── EmployeeAdd.java           # Class for adding employee details
│   ├── EmployeeShow.java          # Class for viewing employee details
│   ├── EmployeeRemove.java        # Class for removing employee records
│   ├── EmployeeUpdate.java        # Class for updating employee details
│   └── CodeExit.java              # Class to handle clean exit of the app
│
├── data/                         # Directory to store employee data files
│   └── file<ID>.txt              # Individual employee data files (e.g., file101.txt)
│
├── docs/                         # Project documentation files
│   ├── EMS_Project_Documentation.pdf
│   └── README.md                 # Project overview and instructions for GitHub
│
└── .gitignore                    # Git ignore file to exclude unnecessary files
```

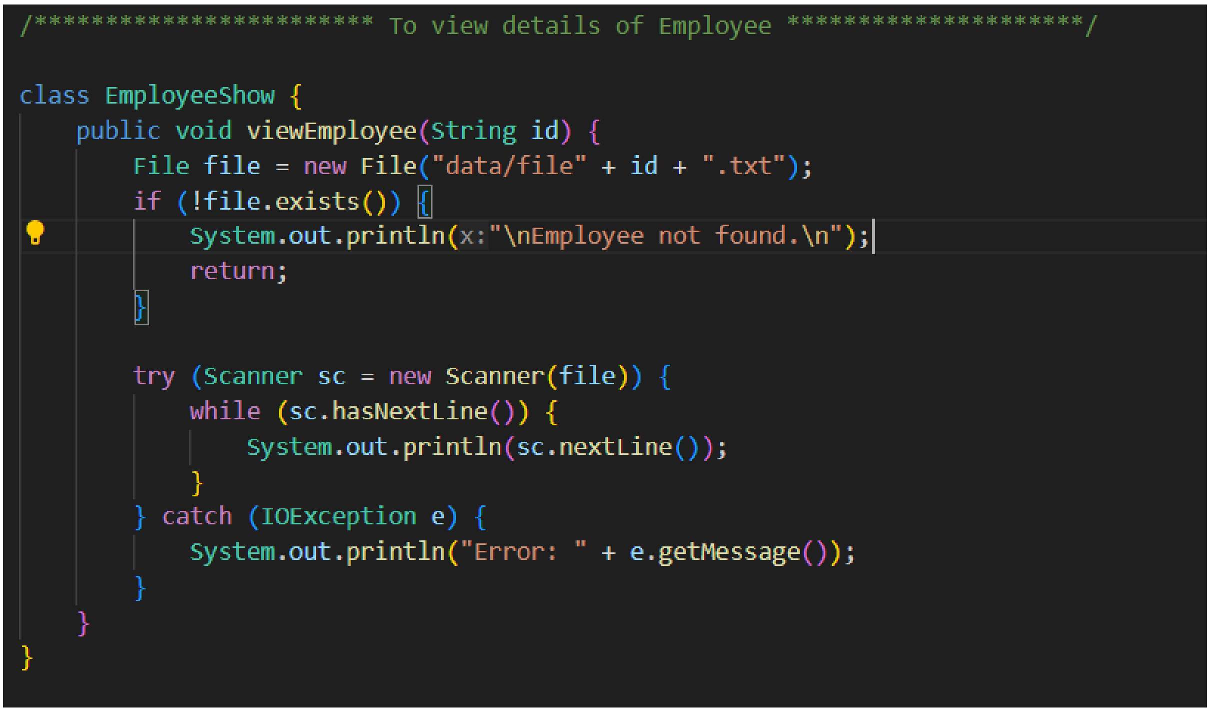
Test & Validation :



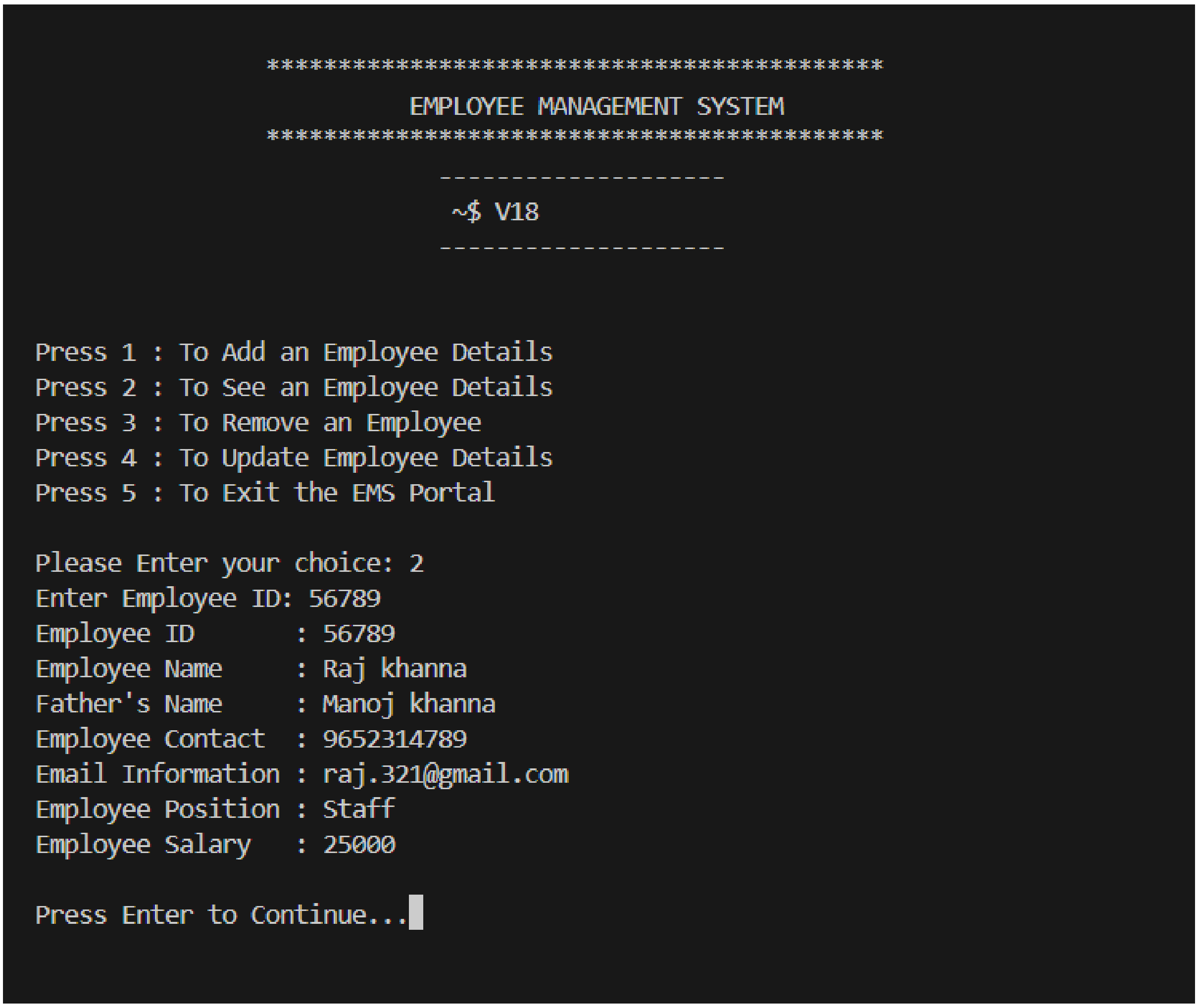
```java
/*********************** To add details of Employee ********************/

class EmployeeAdd {
    public void addEmployee(Scanner sc) {
        Employee emp = new Employee();
        if (!emp.inputDetails(sc)) return;

        File dir = new File(pathname:"data");
        if (!dir.exists()) dir.mkdir();

        File file = new File(dir, "file" + emp.id + ".txt");
        try {
            if (file.createNewFile()) {
                try (FileWriter writer = new FileWriter(file)) {
                    writer.write(emp.toFileString());
                }
                System.out.println(x:"\nEmployee has been added successfully.\n");
            } else {
                System.out.println(x:"\nEmployee already exists.\n");
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```



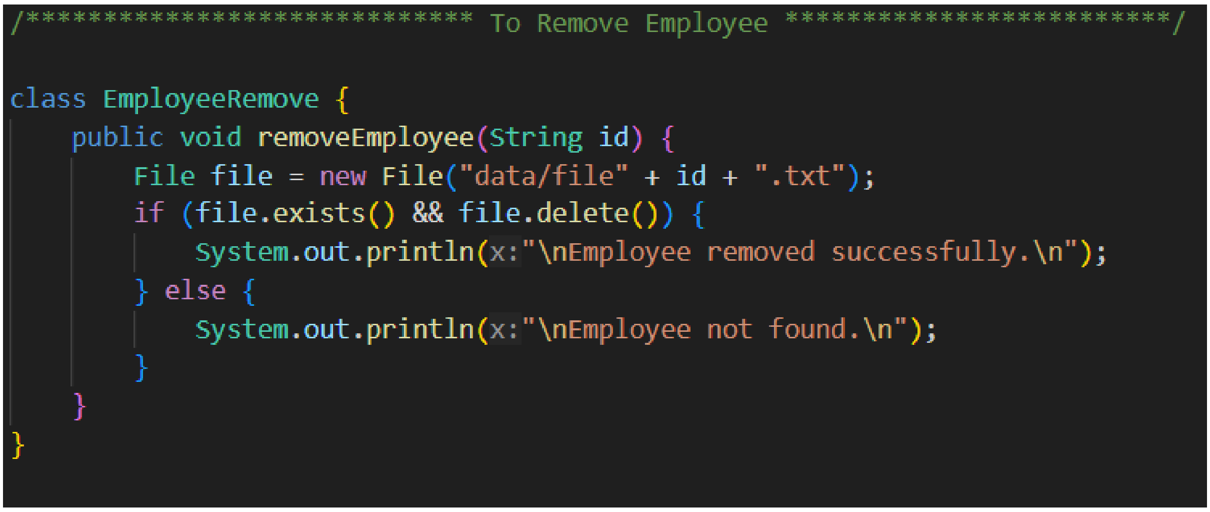
```
                   *******************************************
                             EMPLOYEE MANAGEMENT SYSTEM
                   *******************************************
                               --------------------
                                ~$ V18
                               --------------------


Press 1 : To Add an Employee Details
Press 2 : To See an Employee Details
Press 3 : To Remove an Employee
Press 4 : To Update Employee Details
Press 5 : To Exit the EMS Portal

Please Enter your choice: 1
Enter Employee's ID ----------: 56789
Enter Employee's Name --------: Raj khanna
Enter Father's Name ----------: Manoj khanna
Enter Email ------------------: raj.321@gmail.com
Enter Position ---------------: Staff
Enter Contact ----------------: 9652314789
Enter Salary -----------------: 25000

Employee has been added successfully.


Press Enter to Continue...
```



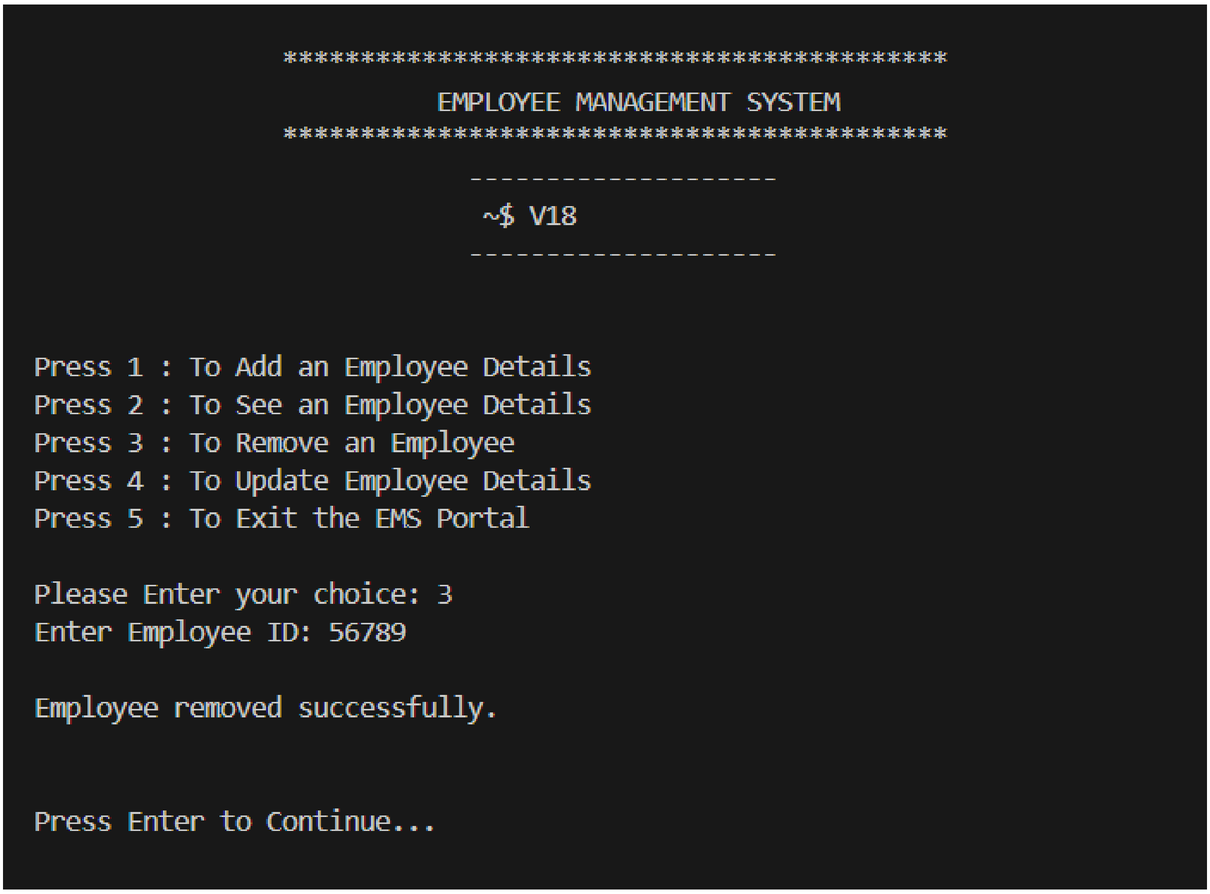
```java
/************************ To view details of Employee ********************/

class EmployeeShow {
    public void viewEmployee(String id) {
        File file = new File("data/file" + id + ".txt");
        if (!file.exists()) {
            System.out.println(x:"\nEmployee not found.\n");
            return;
        }

        try (Scanner sc = new Scanner(file)) {
            while (sc.hasNextLine()) {
                System.out.println(sc.nextLine());
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```



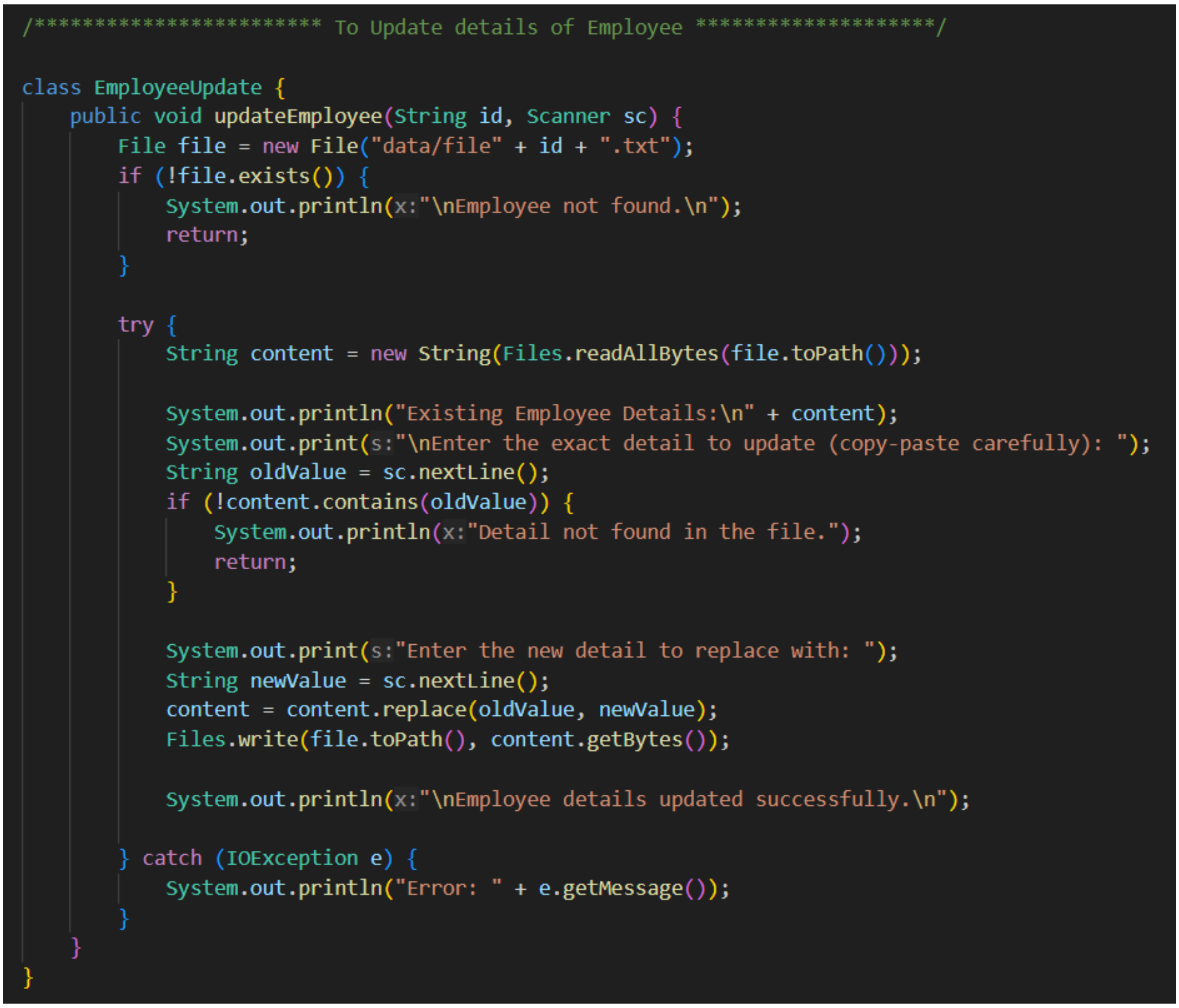
```
                  *******************************************
                            EMPLOYEE MANAGEMENT SYSTEM
                  *******************************************
                              --------------------
                               ~$ V18
                              --------------------


Press 1 : To Add an Employee Details
Press 2 : To See an Employee Details
Press 3 : To Remove an Employee
Press 4 : To Update Employee Details
Press 5 : To Exit the EMS Portal

Please Enter your choice: 2
Enter Employee ID: 56789
Employee ID         : 56789
Employee Name       : Raj khanna
Father's Name       : Manoj khanna
Employee Contact    : 9652314789
Email Information   : raj.321@gmail.com
Employee Position   : Staff
Employee Salary     : 25000

Press Enter to Continue...
```



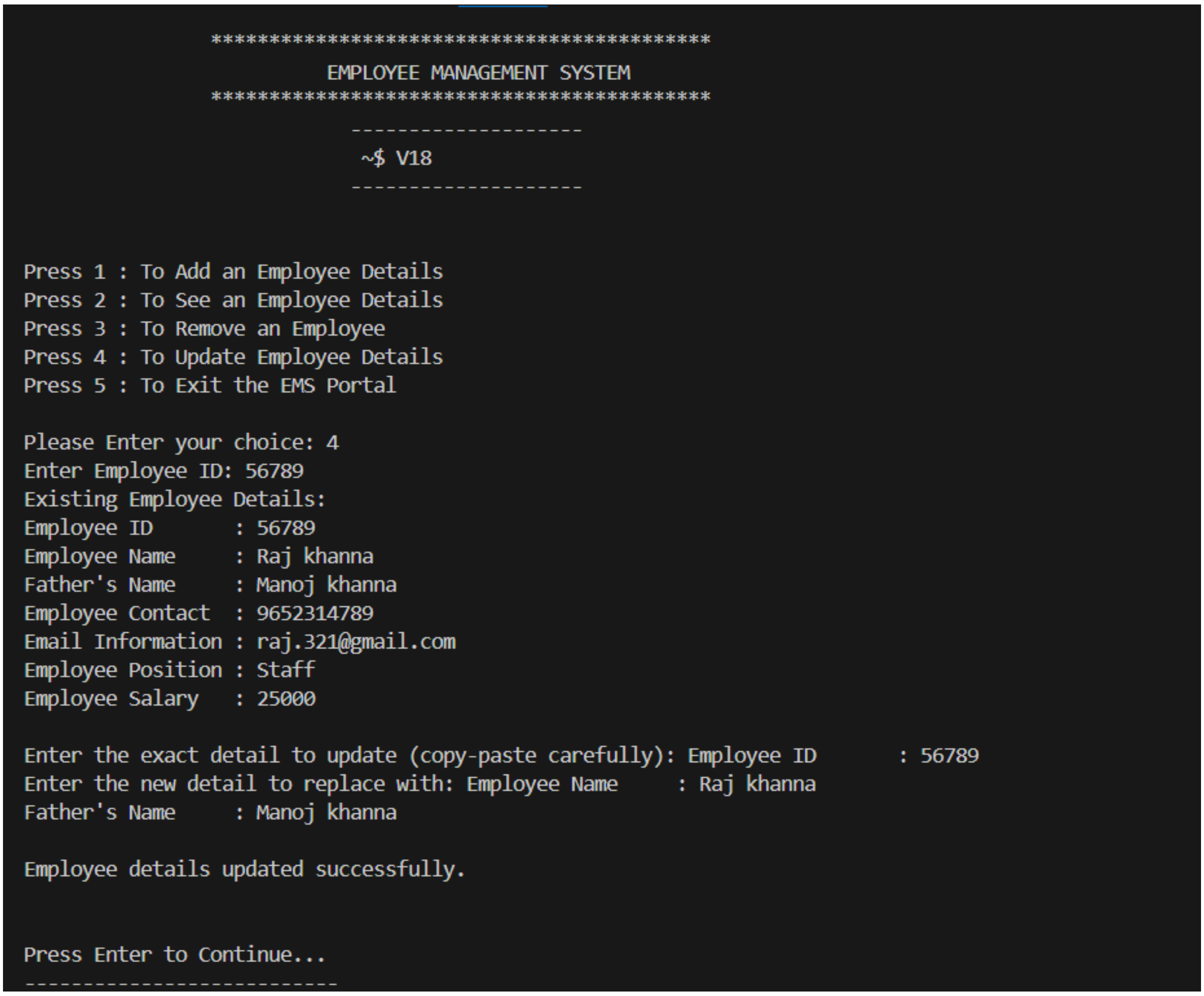
```
/***************************** To Remove Employee *************************/

class EmployeeRemove {
    public void removeEmployee(String id) {
        File file = new File("data/file" + id + ".txt");
        if (file.exists() && file.delete()) {
            System.out.println(x:"\nEmployee removed successfully.\n");
        } else {
            System.out.println(x:"\nEmployee not found.\n");
        }
    }
}
```



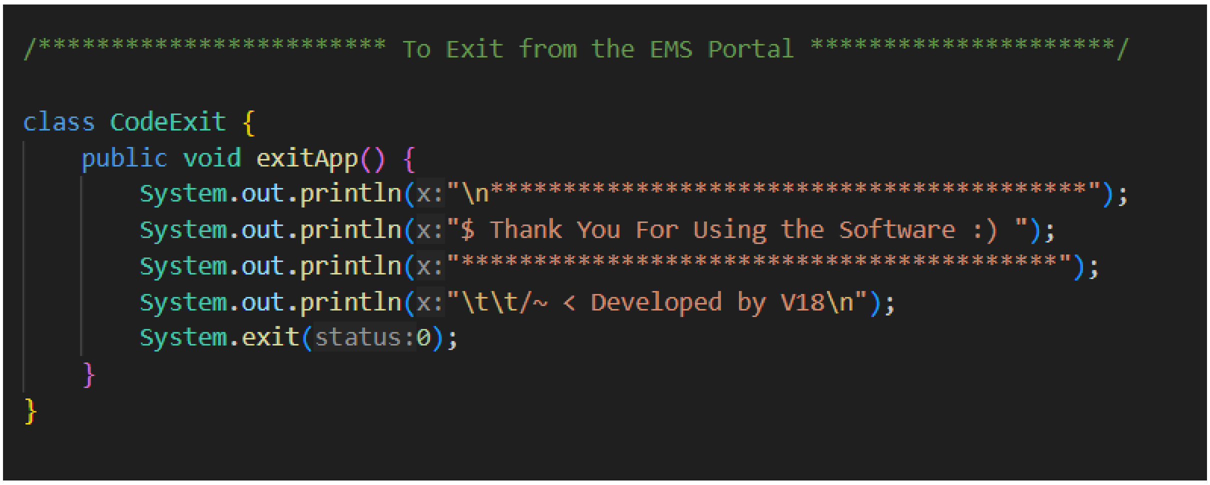
```
                *******************************************

                          EMPLOYEE MANAGEMENT SYSTEM
                *******************************************

                            --------------------
                             ~$ V18
                            --------------------


Press 1 : To Add an Employee Details
Press 2 : To See an Employee Details
Press 3 : To Remove an Employee
Press 4 : To Update Employee Details
Press 5 : To Exit the EMS Portal

Please Enter your choice: 3
Enter Employee ID: 56789

Employee removed successfully.


Press Enter to Continue...
```



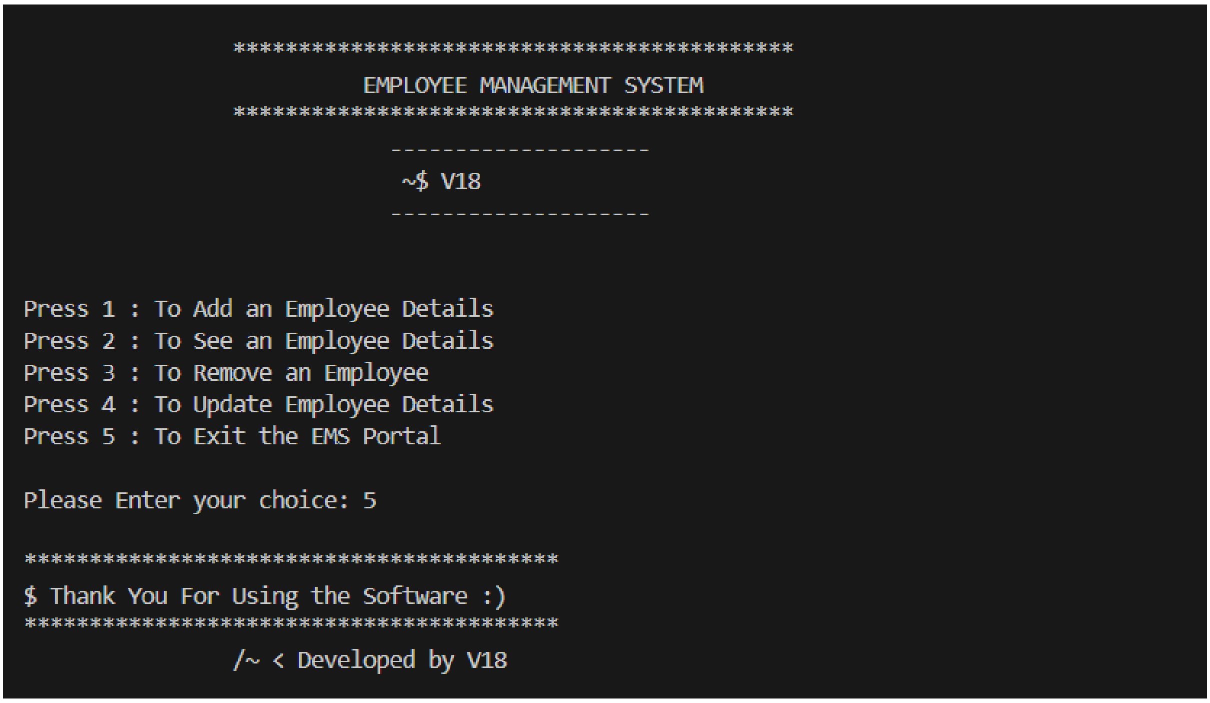
```java
/************************ To Update details of Employee ********************/

class EmployeeUpdate {
    public void updateEmployee(String id, Scanner sc) {
        File file = new File("data/file" + id + ".txt");
        if (!file.exists()) {
            System.out.println(x:"\nEmployee not found.\n");
            return;
        }

        try {
            String content = new String(Files.readAllBytes(file.toPath()));

            System.out.println("Existing Employee Details:\n" + content);
            System.out.print(s:"\nEnter the exact detail to update (copy-paste carefully): ");
            String oldValue = sc.nextLine();
            if (!content.contains(oldValue)) {
                System.out.println(x:"Detail not found in the file.");
                return;
            }

            System.out.print(s:"Enter the new detail to replace with: ");
            String newValue = sc.nextLine();
            content = content.replace(oldValue, newValue);
            Files.write(file.toPath(), content.getBytes());

            System.out.println(x:"\nEmployee details updated successfully.\n");

        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```



```
                ******************************************* 
                          EMPLOYEE MANAGEMENT SYSTEM
                *******************************************
                            --------------------
                             ~$ V18
                            --------------------


Press 1 : To Add an Employee Details
Press 2 : To See an Employee Details
Press 3 : To Remove an Employee
Press 4 : To Update Employee Details
Press 5 : To Exit the EMS Portal

Please Enter your choice: 4
Enter Employee ID: 56789
Existing Employee Details:
Employee ID       : 56789
Employee Name     : Raj khanna
Father's Name     : Manoj khanna
Employee Contact  : 9652314789
Email Information : raj.321@gmail.com
Employee Position : Staff
Employee Salary   : 25000

Enter the exact detail to update (copy-paste carefully): Employee ID       : 56789
Enter the new detail to replace with: Employee Name     : Raj khanna
Father's Name     : Manoj khanna

Employee details updated successfully.


Press Enter to Continue...
---------------------------
```



```java
/************************* To Exit from the EMS Portal **********************/

class CodeExit {
    public void exitApp() {
        System.out.println(x:"\n*******************************************");
        System.out.println(x:"$ Thank You For Using the Software :) ");
        System.out.println(x:"*******************************************");
        System.out.println(x:"\t\t/~ < Developed by V18\n");
        System.exit(status:0);
    }
}
```



```
                *******************************************
                          EMPLOYEE MANAGEMENT SYSTEM
                *******************************************
                            --------------------
                             ~$ V18
                            --------------------


Press 1 : To Add an Employee Details
Press 2 : To See an Employee Details
Press 3 : To Remove an Employee
Press 4 : To Update Employee Details
Press 5 : To Exit the EMS Portal

Please Enter your choice: 5

*****************************************
$ Thank You For Using the Software :)
*****************************************
                /~ < Developed by V18
```

7. □ Conclusion:

This console-based Employee Management System offers a complete, modular solution with essential functionalities and clean error handling with core features, applying techniques to enhance robustness and integration data validation, code quality, and innovation. It's ideal for small businesses or as a learning project for students.