Acknowledgement

We would like to thank "Connect Me," the mobile service provider, for giving us the assignment to create an automated system for handling their customer service personnel information. Their goal of improving efficiency and simplifying HR operations has motivated our team to work hard at developing a solid solution.

Table of Contents

Introduction

Efficiency and accuracy are critical for managing personnel data in today's dynamic business environment, particularly for organisations like "Connect Me" that place a high priority on customer happiness. "Connect Me" has started a quest to automate the management of their customer care personnel information after realising the necessity to modernise their HR procedures.

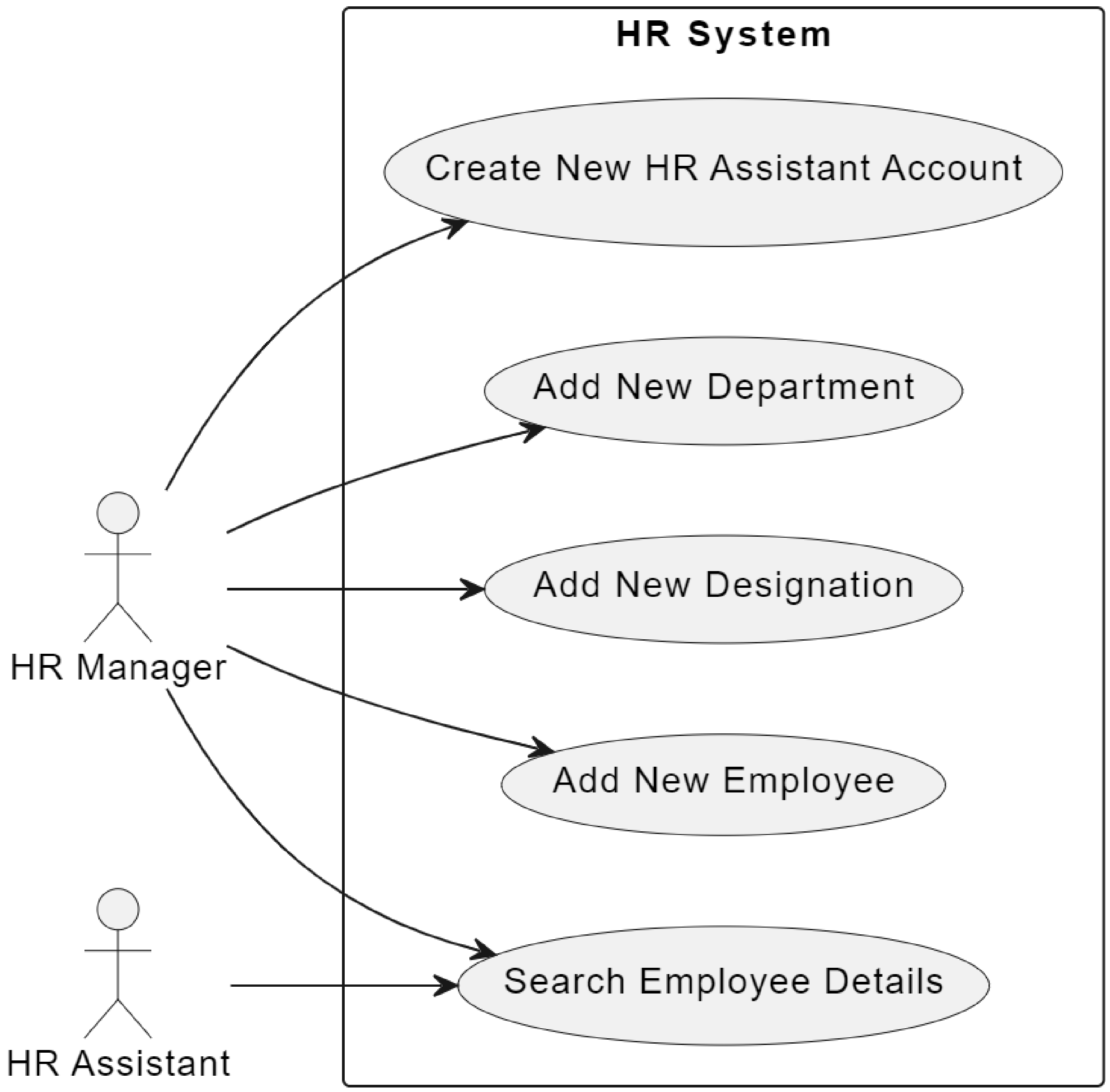
The objective of this programme is to equip HR managers and assistants with the necessary tools to effectively supervise employee data, optimise departmental workflows, and guarantee smooth communication throughout the company.

We want to use the concepts of Object-Oriented Programming (OOP) to create a system that not only satisfies "Connect Me"'s present requirements but also has the scalability and flexibility to grow with the company's needs.

Our project aims to transform "Connect Me"'s workforce management approach, which will ultimately lead to increased output, happier employees, and better customer service.

We cordially invite you to accompany us on this exciting journey as we leverage the power of innovation and technology to transform HR management for "Connect Me."

1 Use Case Diagram:



Explanation:

1. **HR Manager**:

   Add New Department: The HR Manager has the ability to create a new

   Add New Designation: The HR Manager has the ability to add a new title.

   Add New Employee: The HR Manager has the ability to add a new worker and assign them to open positions in departments and job categories.

Search Employee Details: The HR Manager can look up employee information by name, department, and designation.

Create New HR Assistant Account: HR Manager is able to establish a new account for HR Assistant.

HR Assistant:

Search Employee Details: The HR Assistant has the ability to look up employee information by name and department.

1.1 Description of the Use Case Diagram

The "Connect Me" system, which attempts to automate the maintenance of customer care employee details for a mobile service firm, is described in depth in the Use Case Diagram. The HR Manager and HR Assistant roles' functionalities are depicted in the diagram.

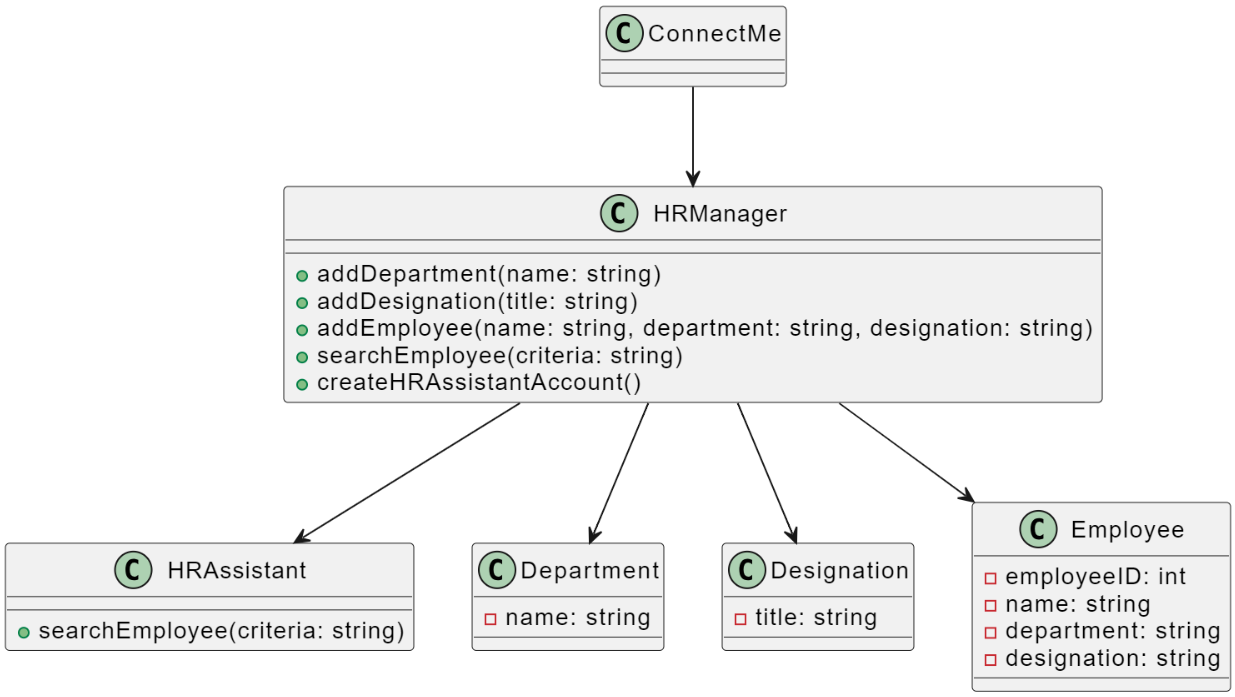
introducing new employees and assigning them to departments and designations, introducing new HR Assistant accounts, searching for employee details using multiple criteria, and adding new departments and designations are all functions available to HR managers.

However, the HR Assistant's access is restricted and she can only look up personnel information by department and name.

To make the tasks and responsibilities of each user level clear and to highlight the system's emphasis on data integrity and security, assumptions are made.

Overall, the application of object-oriented principles and consistent usage of UML notation supports the usage Case Diagram's successful representation of the functionalities of the system. It gives a clear picture of the system's capabilities and is in line with the scenario's goals.

1.2 Class Diagram


ConnectMe
HRManager
addDepartment(name: string)
addDesignation(title: string)
addEmployee(name: string, department: string, designation: string)
searchEmployee(criteria: string)
createHRAssistantAccount()
HRAssistant
searchEmployee(criteria: string)
Department
name: string
Designation
title: string
Employee
employeeID: int
name: string
department: string
designation: string

Explanation:

- **ConnectMe**: Represents the system as a whole.

- **HRManager**: Class responsible for managing departments, designations, employees, and HR Assistant accounts.
  - addDepartment(name: string): Method to add a new department.
  - addDesignation(title: string): Method to add a new designation.
  - addEmployee(name: string, department: string, designation: string): Method to add a new employee.
  - searchEmployee(criteria: string): Method to search for employees based on criteria (department, designation, name).
  - createHRAssistantAccount(): Method to create a new HR Assistant account.

- **HRAssistant**: Class responsible for searching employee details.
    - searchEmployee(criteria: string): Method to search for employees based on criteria (department, name).
- **Department**: Represents a department entity with a name attribute.
- **Designation**: Represents a designation entity with a title attribute.
- **Employee**: Represents an employee entity with attributes such as employeeID, name, department, and designation.

1.2.1 Description of the Class Diagram

A thorough description of the "ConnectMe" system's main elements, characteristics, and operations is given in the Class Diagram.

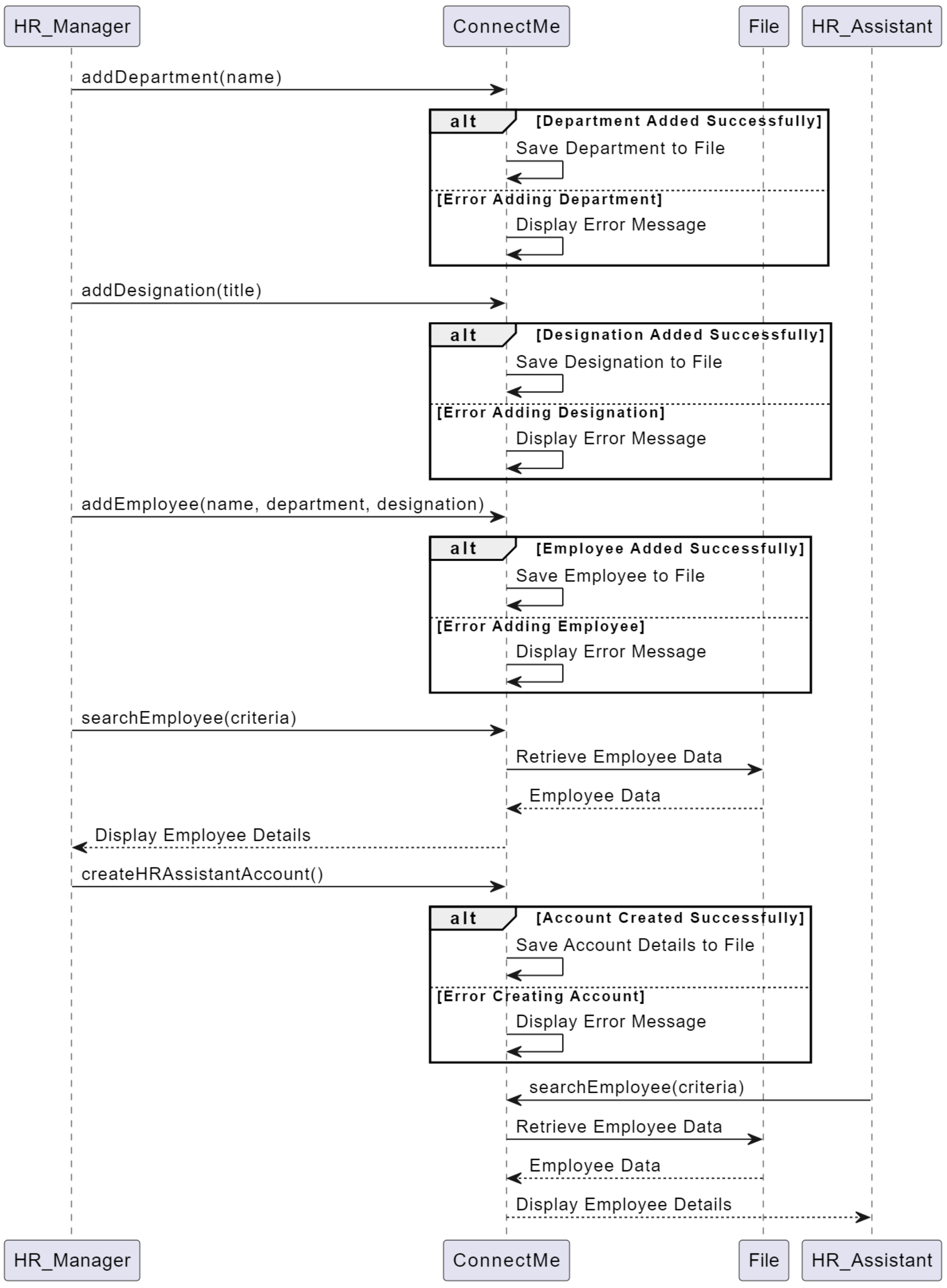
- **ConnectMe**: Represents the system as a whole.
- **HRManager**: Responsible for managing departments, designations, employees, and HR Assistant accounts.
- **HRAssistant**: Responsible for searching employee details.
- **Department**: Represents a department entity.
- **Designation**: Represents a designation entity.
- **Employee**: Represents an employee entity.

Clearness on the behaviour and scope of the system, including employee IDs that are unique and data integrity safeguards, is ensured by assumptions.

The explanation emphasises how well the diagram conforms to UML standards and how well it illustrates object-oriented principles.

All things considered, the Class Diagram effectively directs the development process by acting as a fundamental blueprint for putting the system's functionalities into code.

1.3. Sequence Diagram



Explanation:

The relationships between the actors (HR Manager and HR Assistant) and the ConnectMe system for different procedures are depicted in the Sequence Diagram.

- **Add Department**: HR Manager initiates the addition of a new department. If successful, ConnectMe saves the department to a file; otherwise, it displays an error message.
- **Add Designation**: HR Manager adds a new designation. ConnectMe saves the designation to a file upon successful addition; otherwise, it displays an error message.
- **Add Employee**: HR Manager adds a new employee and allocates them to a department and designation. ConnectMe saves the employee details to a file upon successful addition; otherwise, it displays an error message.
- **Search Employee**: HR Manager searches for employee details based on specified criteria. ConnectMe retrieves employee data from the file, displays it, and sends it back to HR Manager.
- **Create HR Assistant Account**: HR Manager creates a new HR Assistant account. ConnectMe saves the account details to a file if created successfully; otherwise, it displays an error message.
- **Search Employee (HR Assistant)**: HR Assistant searches for employee details based on specified criteria. ConnectMe retrieves employee data from the file, displays it, and sends it back to HR Assistant.

The flow of control throughout various processes, such as error handling mechanisms and data retrieval from the file system, is highlighted in this comprehensive sequence of interactions.

1.3 Description of the Sequence Diagram

The relationship between the HR Manager, HR Assistant, Connect Me system, and File system is depicted in the Sequence Diagram.

The HR Manager starts things like adding personnel, departments, and designations; it also searches for employee information and creates HR Assistant accounts. These requests are handled by the Connect Me system, which communicates with the File system to save and retrieve data. Branches for error handling are integrated to address possible malfunctions in every process.

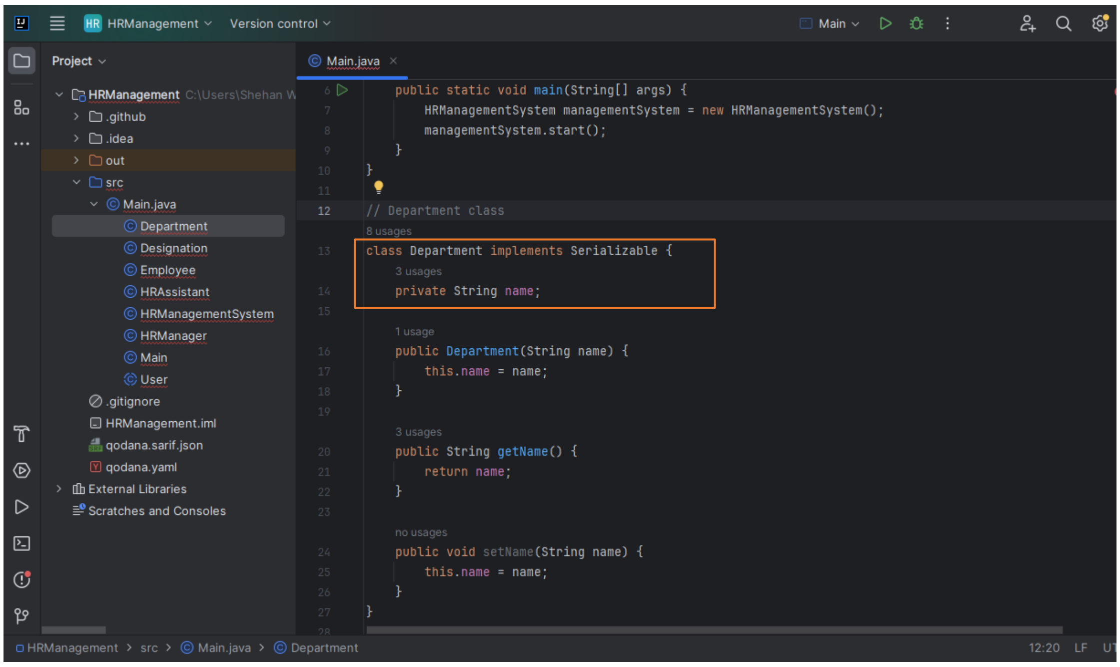
Regarding data storage and error handling in the Connect Me system, assumptions are made. The series of interactions is successfully captured in the diagram, illustrating the behaviour of the system and the flow of control during different processes. It offers a thorough understanding of the functionality of the system and supports design choices. Analysing critically guarantees that the goals of the scenario are met.

2. Explain the proper use of object-oriented concepts.

I'll use the proper object-oriented ideas to create the system based on the scenario that has been provided. Let's outline the functionalities of each class and dissect the system into smaller units.

2.1. Department.java

* This class represents a department within the company.
* It encapsulates the department name.
* It demonstrates the principle of encapsulation by providing private access to the department name and using public methods to manipulate it.



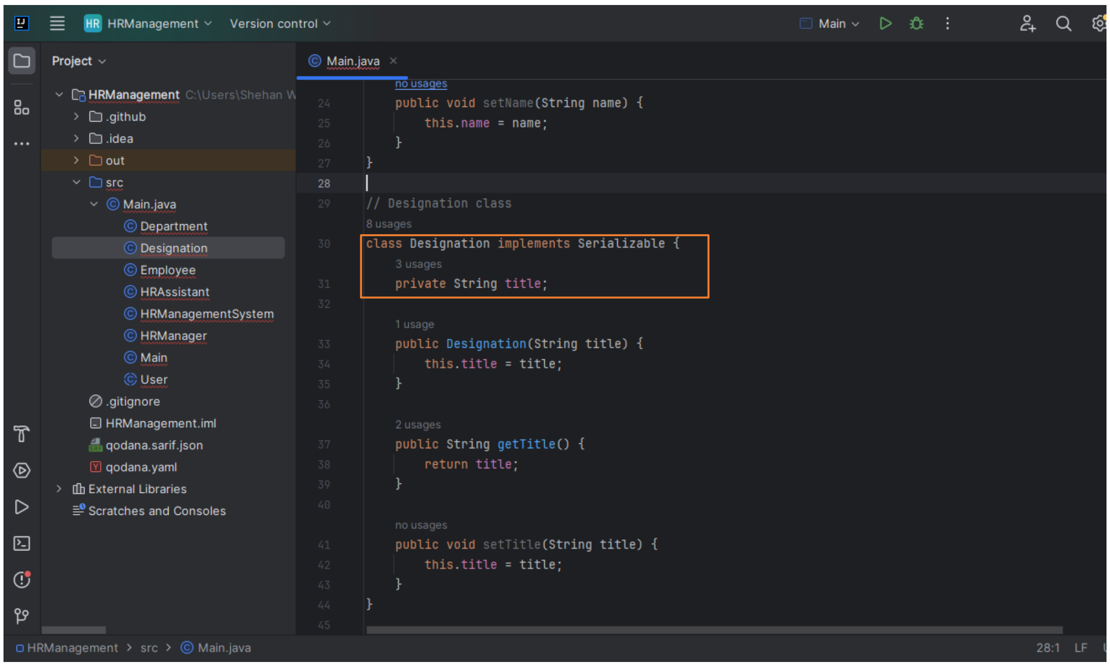
```java
    public static void main(String[] args) {
        HRManagementSystem managementSystem = new HRManagementSystem();
        managementSystem.start();
    }
}

// Department class
8 usages
class Department implements Serializable {
    3 usages
    private String name;

    1 usage
    public Department(String name) {
        this.name = name;
    }

    3 usages
    public String getName() {
        return name;
    }

    no usages
    public void setName(String name) {
        this.name = name;
    }
}
```

2.2. Designation.java

* This class represents a job designation within the company.

* Similar to the Department class, it encapsulates the job designation title.

* It also demonstrates encapsulation



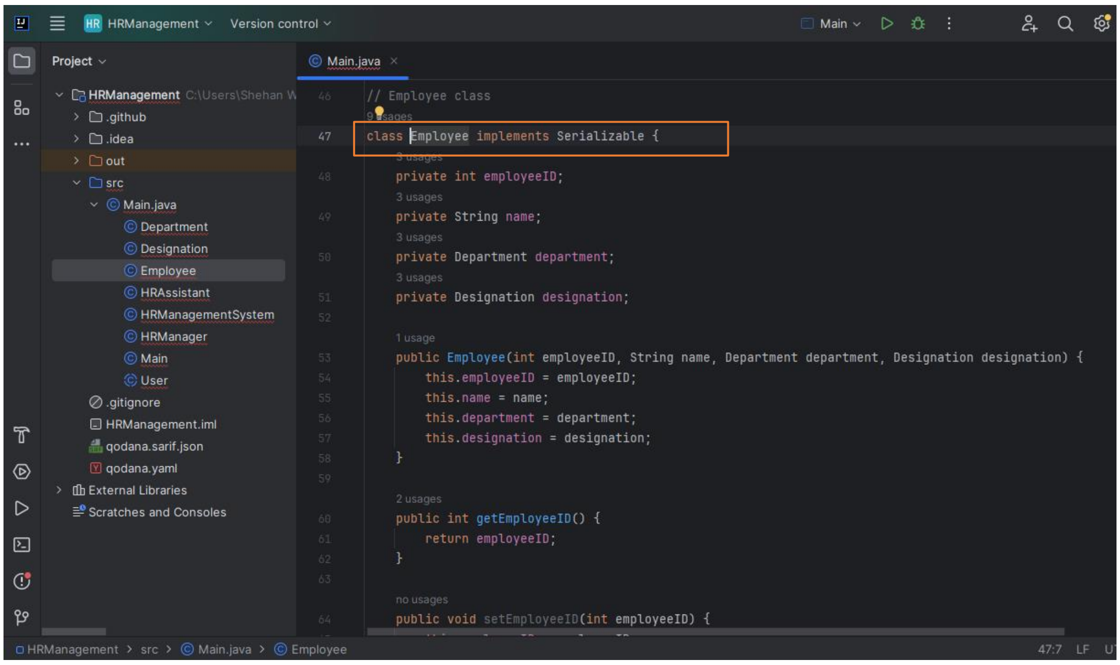
```java
    public void setName(String name) {
        this.name = name;
    }
}

// Designation class
class Designation implements Serializable {
    private String title;

    public Designation(String title) {
        this.title = title;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
}
```

2.3. Employee.java

* This class represents an employee within the company.

* It encapsulates employee details such as ID, name, department, and designation.

* It demonstrates encapsulation through private access to employee details and provides public methods for accessing and modifying them.



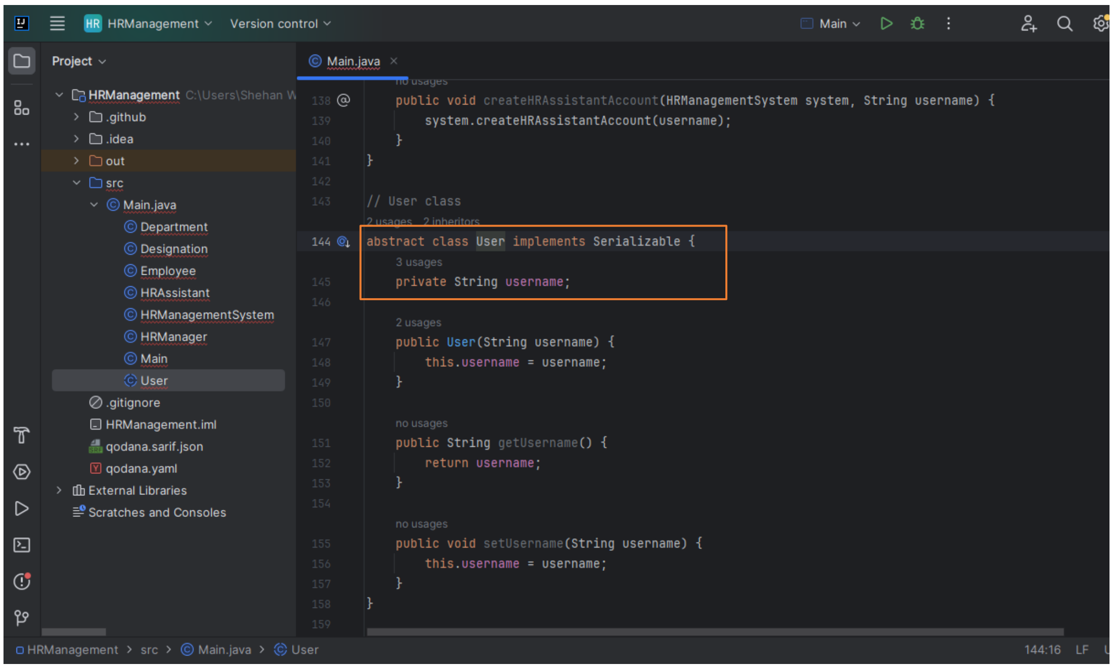
```java
// Employee class
class Employee implements Serializable {
    private int employeeID;
    private String name;
    private Department department;
    private Designation designation;

    public Employee(int employeeID, String name, Department department, Designation designation) {
        this.employeeID = employeeID;
        this.name = name;
        this.department = department;
        this.designation = designation;
    }

    public int getEmployeeID() {
        return employeeID;
    }

    public void setEmployeeID(int employeeID) {
```

2.4. User.java

* This abstract class represents a user in the system.

* It encapsulates the username.

* It demonstrates abstraction by defining common attributes and behaviors for users without specifying implementation details.



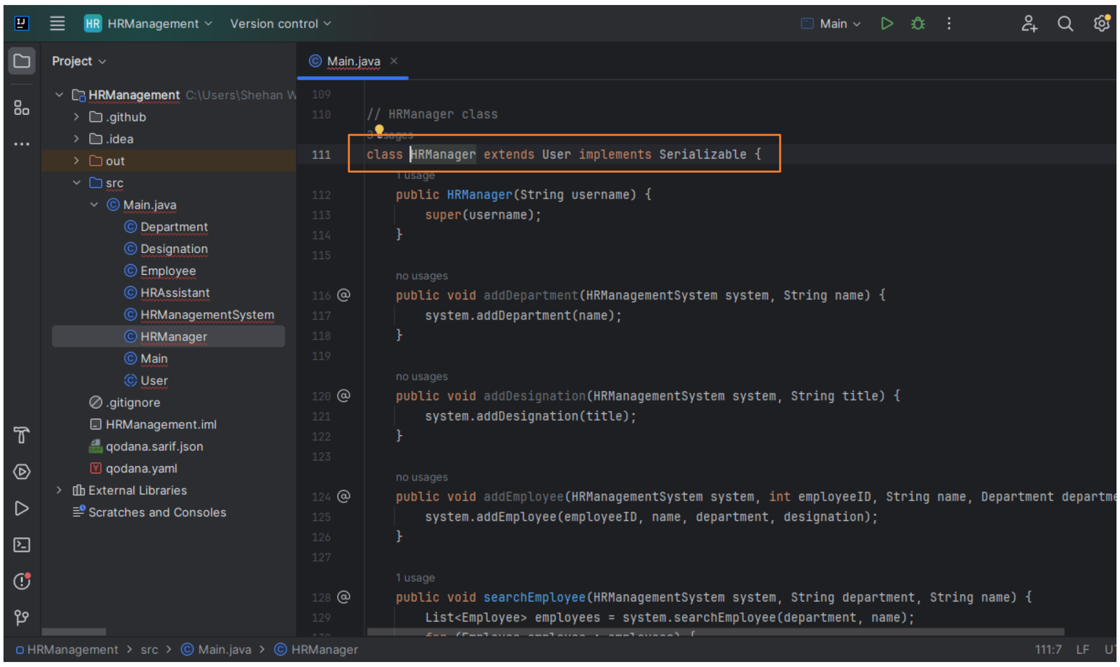
```
    no usages
    public void createHRAssistantAccount(HRManagementSystem system, String username) {
        system.createHRAssistantAccount(username);
    }
}

// User class
2 usages   2 inheritors
abstract class User implements Serializable {
    3 usages
    private String username;

    2 usages
    public User(String username) {
        this.username = username;
    }

    no usages
    public String getUsername() {
        return username;
    }

    no usages
    public void setUsername(String username) {
        this.username = username;
    }
}
```

2.5. HRManager.java

* This class also extends the User class and represents an HR Manager.

* It contains methods to perform various HR management tasks such as adding departments, designations, employees, searching employees, and creating HR assistant accounts.

* It also demonstrates inheritance by extending the User class.



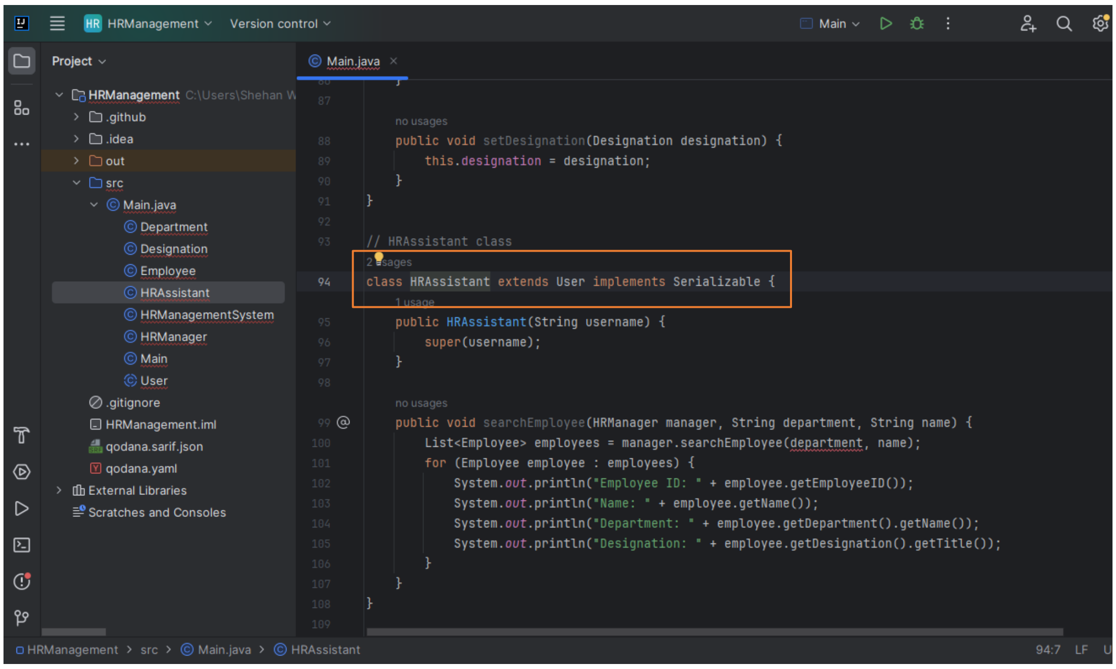
```
// HRManager class
3 usages
class HRManager extends User implements Serializable {
    1 usage
    public HRManager(String username) {
        super(username);
    }

    no usages
    public void addDepartment(HRManagementSystem system, String name) {
        system.addDepartment(name);
    }

    no usages
    public void addDesignation(HRManagementSystem system, String title) {
        system.addDesignation(title);
    }

    no usages
    public void addEmployee(HRManagementSystem system, int employeeID, String name, Department departme
        system.addEmployee(employeeID, name, department, designation);
    }

    1 usage
    public void searchEmployee(HRManagementSystem system, String department, String name) {
        List<Employee> employees = system.searchEmployee(department, name);
```

2.6. HRAssistant.java

* This class extends the User class and represents an HR Assistant.

* It contains a method searchEmployee() to search for employee details.

* It utilizes inheritance by extending the User class



```
    no usages
    public void setDesignation(Designation designation) {
        this.designation = designation;
    }
}

// HRAssistant class
2 usages
class HRAssistant extends User implements Serializable {
    1 usage
    public HRAssistant(String username) {
        super(username);
    }

    no usages
    public void searchEmployee(HRManager manager, String department, String name) {
        List<Employee> employees = manager.searchEmployee(department, name);
        for (Employee employee : employees) {
            System.out.println("Employee ID: " + employee.getEmployeeID());
            System.out.println("Name: " + employee.getName());
            System.out.println("Department: " + employee.getDepartment().getName());
            System.out.println("Designation: " + employee.getDesignation().getTitle());
        }
    }
}
```

2.7. HRManagementSystem.java

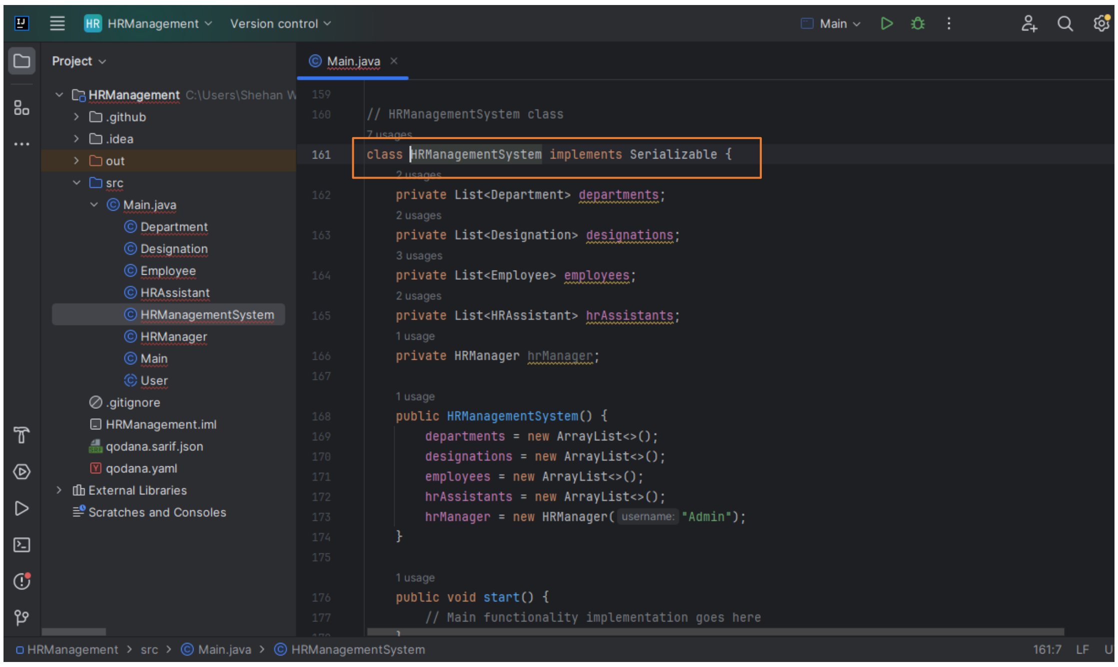
* This class represents the main HR management system.

* It encapsulates various lists of departments, designations, employees, and HR assistants.

* It contains methods for adding departments, designations, employees, searching employees, and creating HR assistant accounts.

* It demonstrates encapsulation by encapsulating data within the system and providing methods to manipulate it.

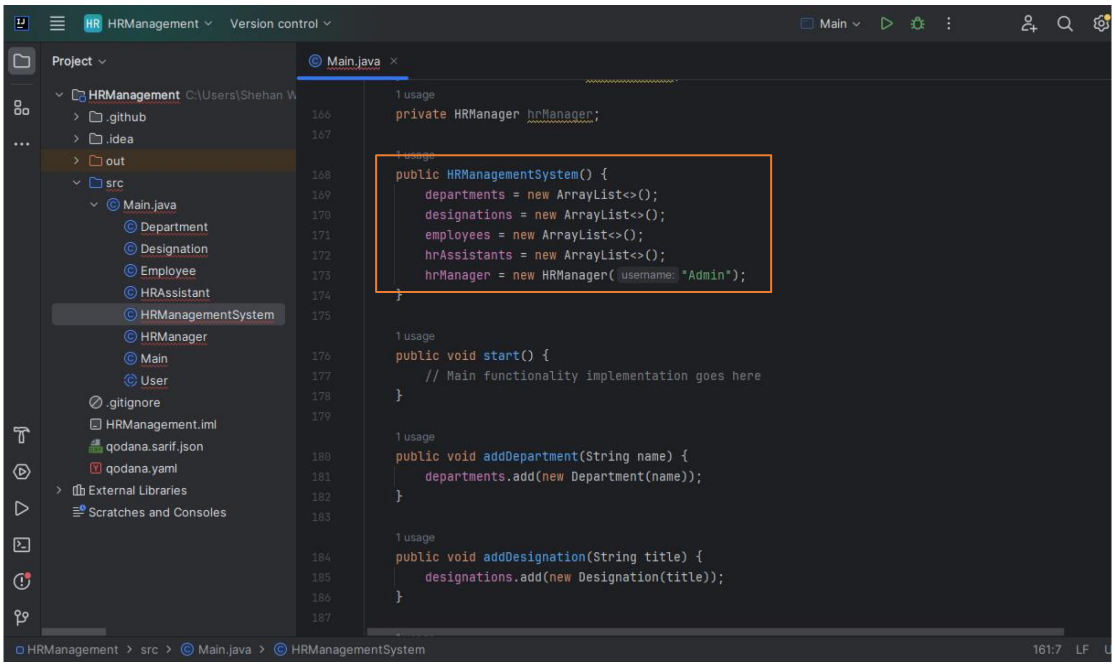
* It also demonstrates polymorphism by allowing different types of users (HRManager and HRAssistant) to perform actions based on their roles.



```java
// HRManagementSystem class
class HRManagementSystem implements Serializable {
    private List<Department> departments;
    private List<Designation> designations;
    private List<Employee> employees;
    private List<HRAssistant> hrAssistants;
    private HRManager hrManager;

    public HRManagementSystem() {
        departments = new ArrayList<>();
        designations = new ArrayList<>();
        employees = new ArrayList<>();
        hrAssistants = new ArrayList<>();
        hrManager = new HRManager( username: "Admin");
    }

    public void start() {
        // Main functionality implementation goes here
```

2.8 Object

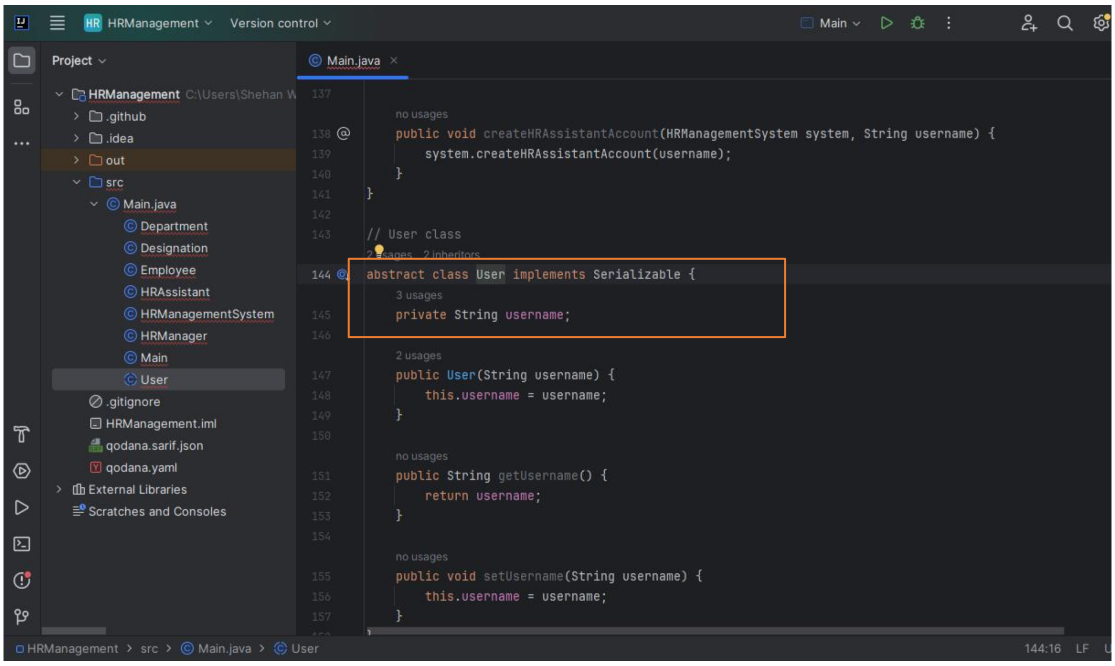
The objects in this code are instances of lists, or ArrayLists, which are used to store information about users, staff members, departments, and titles. It is implied by the UI instance's construction and presentation that the purpose of this graphical user interface is to allow users to interact with the HR management system.



```java
    1 usage
    private HRManager hrManager;

    1 usage
    public HRManagementSystem() {
        departments = new ArrayList<>();
        designations = new ArrayList<>();
        employees = new ArrayList<>();
        hrAssistants = new ArrayList<>();
        hrManager = new HRManager( username: "Admin");
    }

    1 usage
    public void start() {
        // Main functionality implementation goes here
    }

    1 usage
    public void addDepartment(String name) {
        departments.add(new Department(name));
    }

    1 usage
    public void addDesignation(String title) {
        designations.add(new Designation(title));
    }
```

2.9 Abstraction

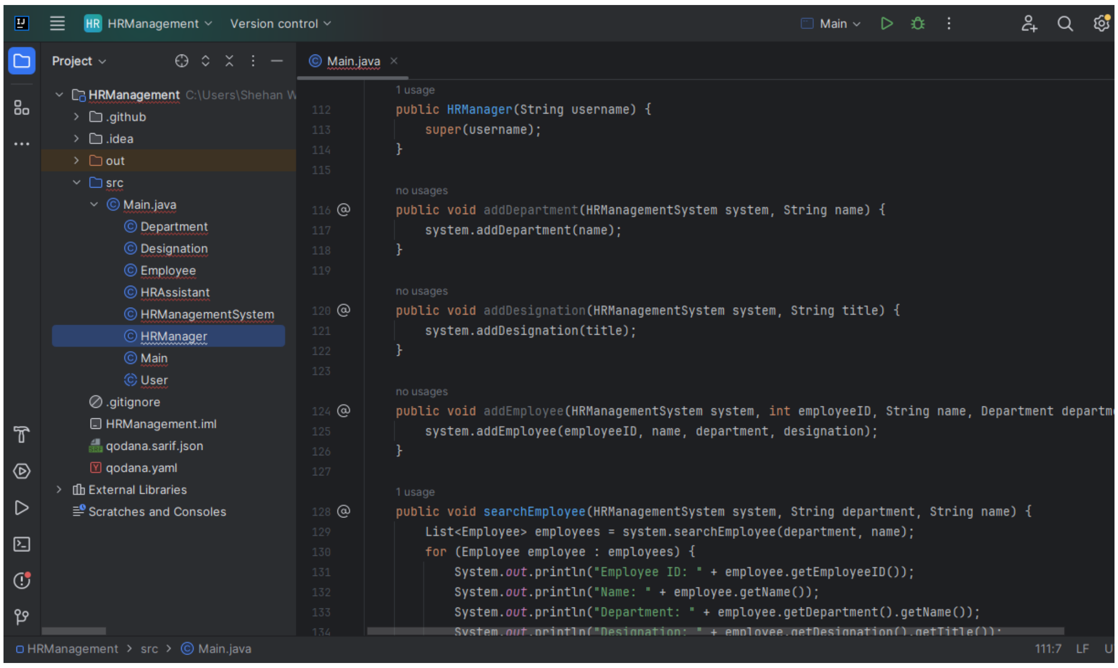
Abstract classes like User provide a common interface for concrete subclasses (HRManager and HRAssistant) to implement.



```java
    no usages
    public void createHRAssistantAccount(HRManagementSystem system, String username) {
        system.createHRAssistantAccount(username);
    }
}

// User class
2 usages  2 inheritors
abstract class User implements Serializable {
    3 usages
    private String username;

    2 usages
    public User(String username) {
        this.username = username;
    }

    no usages
    public String getUsername() {
        return username;
    }

    no usages
    public void setUsername(String username) {
        this.username = username;
    }
```

2.10 Inheritance

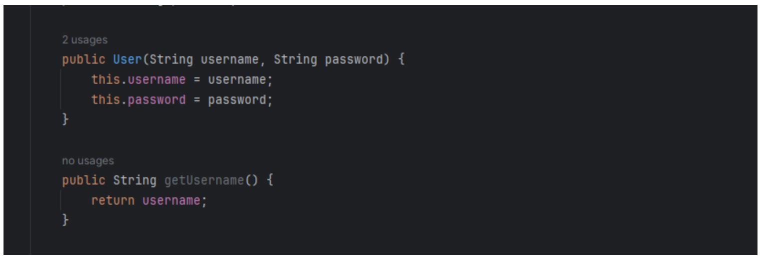
HRManager and HRAssistant extend the User class to inherit common attributes and behaviors.



```java
1 usage
public HRManager(String username) {
    super(username);
}

no usages
public void addDepartment(HRManagementSystem system, String name) {
    system.addDepartment(name);
}

no usages
public void addDesignation(HRManagementSystem system, String title) {
    system.addDesignation(title);
}

no usages
public void addEmployee(HRManagementSystem system, int employeeID, String name, Department departm
    system.addEmployee(employeeID, name, department, designation);
}

1 usage
public void searchEmployee(HRManagementSystem system, String department, String name) {
    List<Employee> employees = system.searchEmployee(department, name);
    for (Employee employee : employees) {
        System.out.println("Employee ID: " + employee.getEmployeeID());
        System.out.println("Name: " + employee.getName());
        System.out.println("Department: " + employee.getDepartment().getName());
        System.out.println("Designation: " + employee.getDesignation().getTitle());
```

2.11 Encapsulation

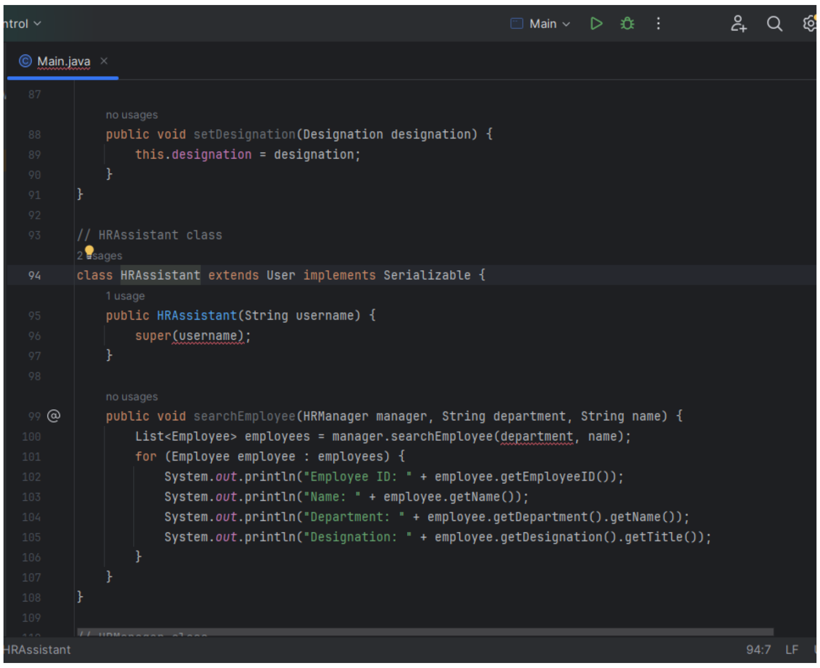
Each class encapsulates its data and provides public methods for interaction, ensuring data integrity and modularity.



```java
2 usages
public User(String username, String password) {
    this.username = username;
    this.password = password;
}

no usages
public String getUsername() {
    return username;
}
```

2.12 Polymorphism

Different types of users (HRManager and HRAssistant) can be treated uniformly using the User superclass, allowing for flexible code design and reusability.



```java
        public void setDesignation(Designation designation) {
            this.designation = designation;
        }
    }

    // HRAssistant class
    class HRAssistant extends User implements Serializable {
        public HRAssistant(String username) {
            super(username);
        }

        public void searchEmployee(HRManager manager, String department, String name) {
            List<Employee> employees = manager.searchEmployee(department, name);
            for (Employee employee : employees) {
                System.out.println("Employee ID: " + employee.getEmployeeID());
                System.out.println("Name: " + employee.getName());
                System.out.println("Department: " + employee.getDepartment().getName());
                System.out.println("Designation: " + employee.getDesignation().getTitle());
            }
        }
    }
```

These classes follow the tenets of object-oriented programming, including polymorphism, inheritance, and encapsulation. I've also included serialisation so you can store and retrieve data from a file. Because each class has a distinct duty, the system is modular and simpler to manage. The primary control point of the system is the HRManagementSystem class.

.

3.User Manual for HR Management System

3.1 Introduction:

Greetings from the "Connect Me" mobile service company's HR management system. The HR manager and assistant's task of maintaining employee details is automated by this technology. It enables the addition of new divisions, titles, and staff members as well as the search for personnel information using a variety of filters.

3.2 System Requirements:

Operating System: Windows 10 or later, macOS, Linux

Java Runtime Environment (JRE) version 8 or later installed on the system.

3.3 Installation:

Download the HR Management System JAR file from the provided link.

Double-click the JAR file to launch the application.

3.4 User Levels and Functionalities:

HR Manager:

Add New Departments and Designations

Add New Employees

Search Employee Details

Create a New Account (HR Assistant)

HR Assistant:

Search Employee Details

3.5 User Manual

Accessing the HR Manager Account

3.5.1. Enter the correct password.

Steps	• Choose the accurate user name. • Kindly input the accurate password. • Press the "Login" button.

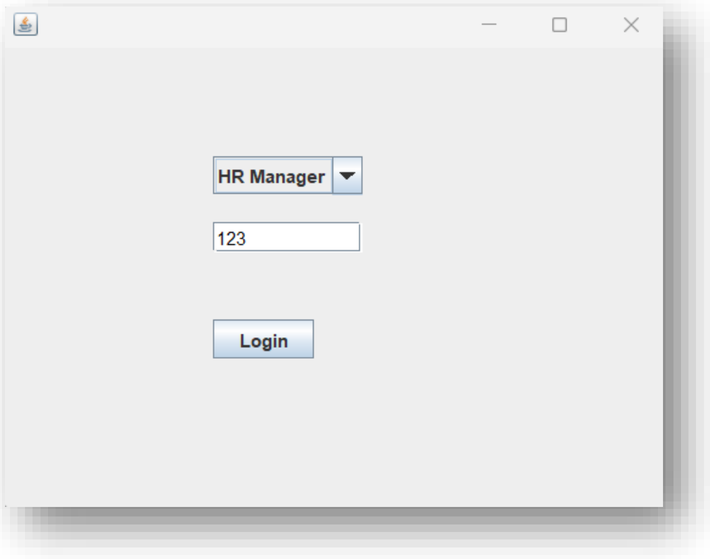
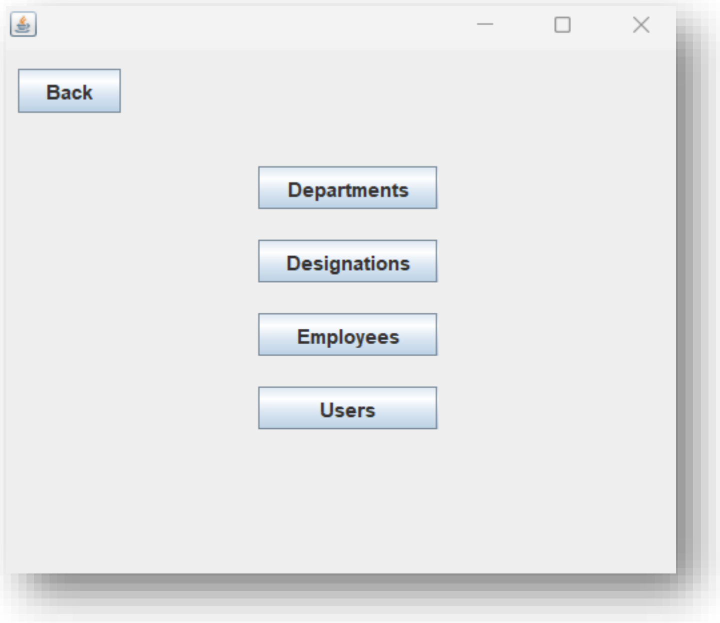
	
Result	Entering the HR Manager account successfully
	

Table 1: Type the right password here

3.5.2 Put in the wrong password.

Steps	*Choose the accurate user name. *Make a mistake in the password entry. *Press the "Login" button.

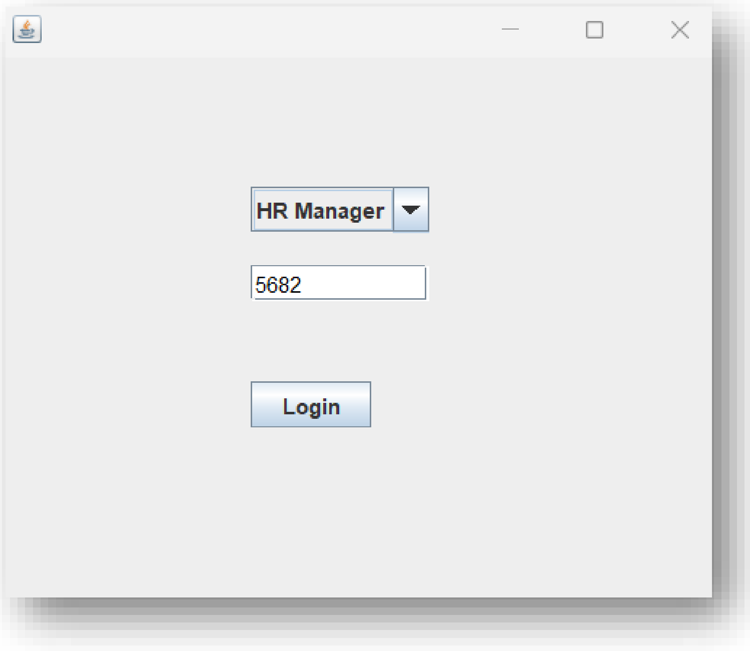
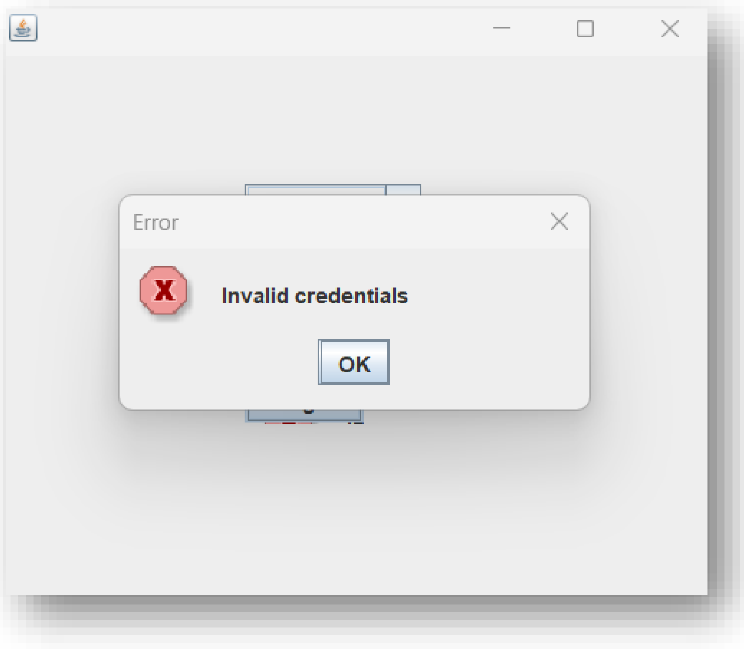
	 HR Manager 5682 Login
Result	An error notice shows up.
	 Error Invalid credentials OK

Table 1:Enter the incorrect password

3.5.3 Adding the new departments

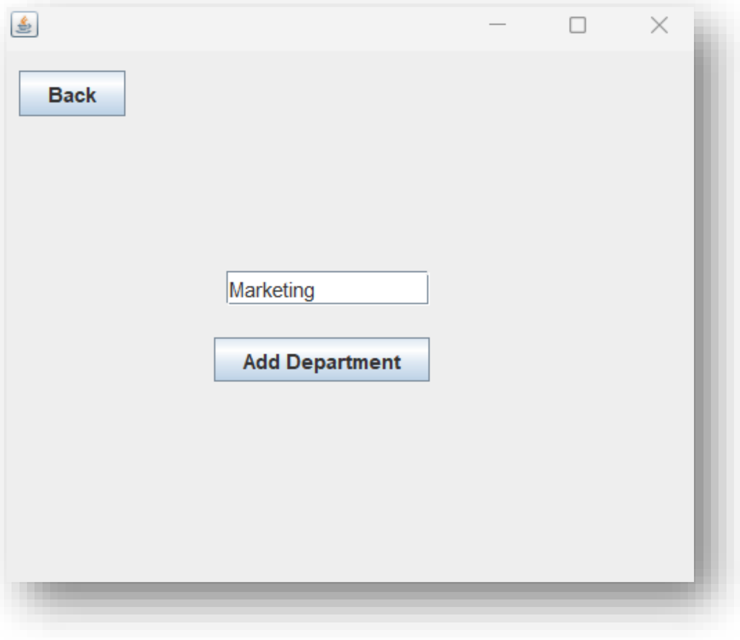
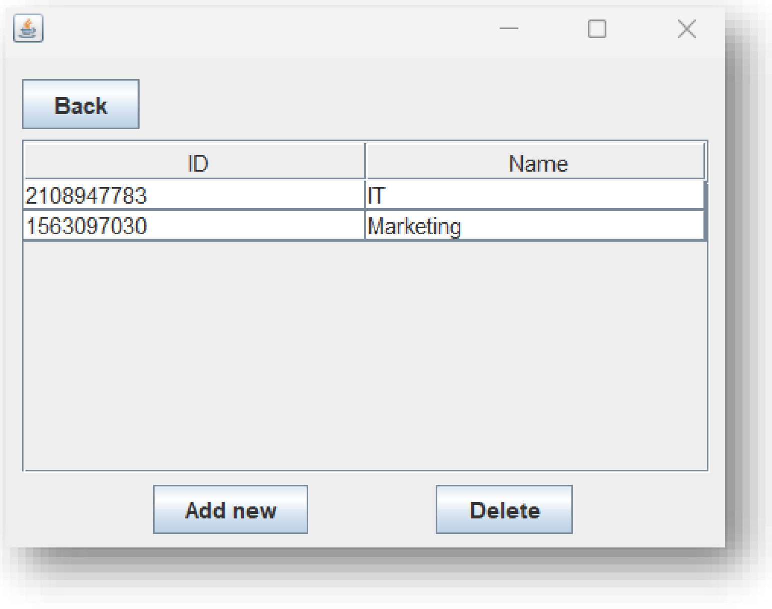
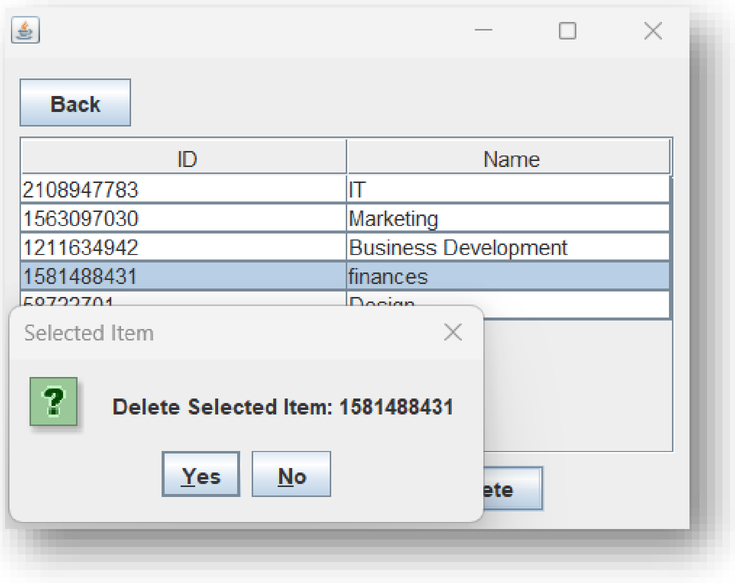
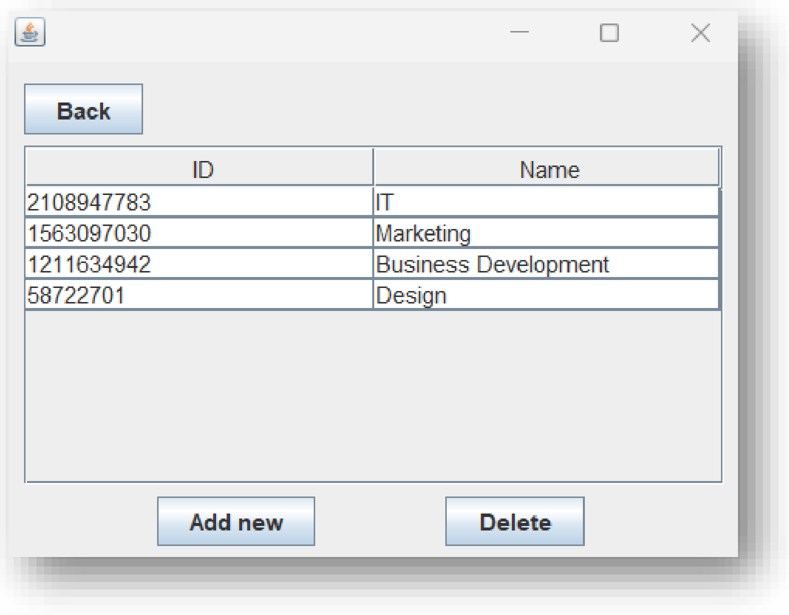
Steps	*Press the "Department" icon. *Select "Add new" from the menu. *Type in the name of the department. *Select "Add Department" from the menu.
	 Marketing
Result	The department was successfully added.
	 ID / Name: 2108947783 / IT; 1563097030 / Marketing

Table 2: Including the newly created departments

3.5.4 Remove the departments.

Steps	*To eliminate a column, select it. *Select the "Delete" option. *Choose between the "Yes" and "No" buttons.
	 ID | Name 2108947783 | IT 1563097030 | Marketing 1211634942 | Business Development 1581488431 | finances 58722701 | Design Selected Item Delete Selected Item: 1581488431 Back · Yes · No
Result	The department was effectively eliminated.

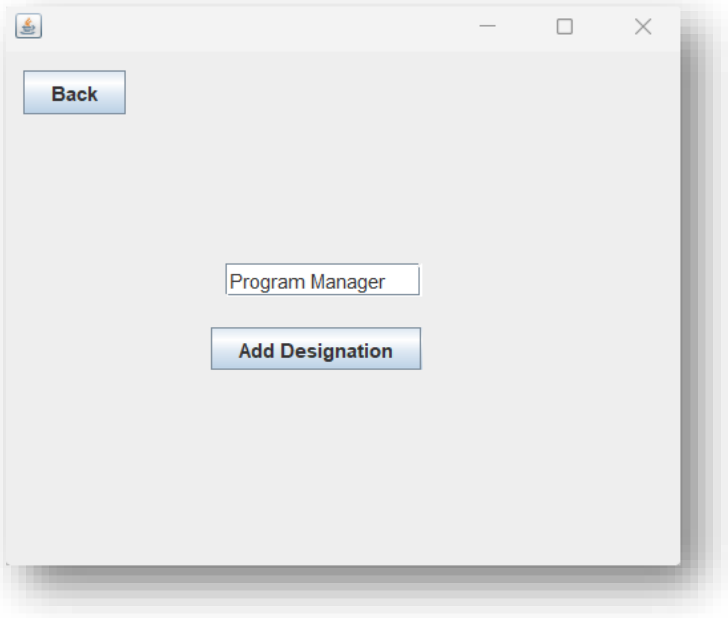
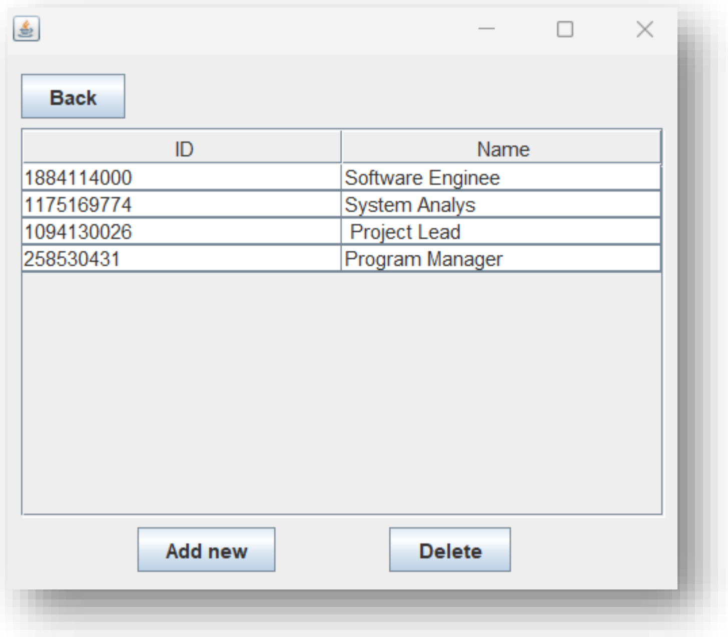
	

ID	Name
2108947783	IT
1563097030	Marketing
1211634942	Business Development
58722701	Design

Table 3: Eliminate the departments.

3.5. 5 Including the newly assigned titles

Steps	*Press the "Designations" icon. *Select "Add new" from the menu. *Type in the name of the designation. *Select "Add Designation" from the menu.

	 Program Manager
Result	The addition of the appellation was accomplished with success.
	

ID	Name
1884114000	Software Enginee
1175169774	System Analys
1094130026	Project Lead
258530431	Program Manager

Table 4: Including the newly assigned titles

3.5.6 Employee admission based on appropriate department and classification

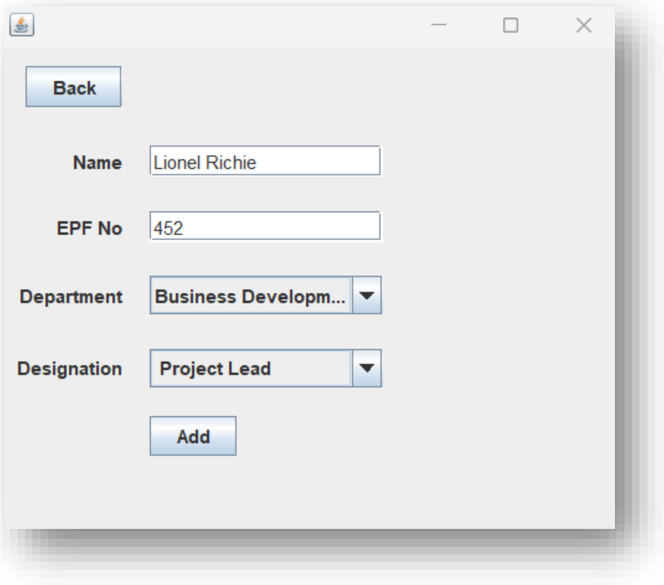
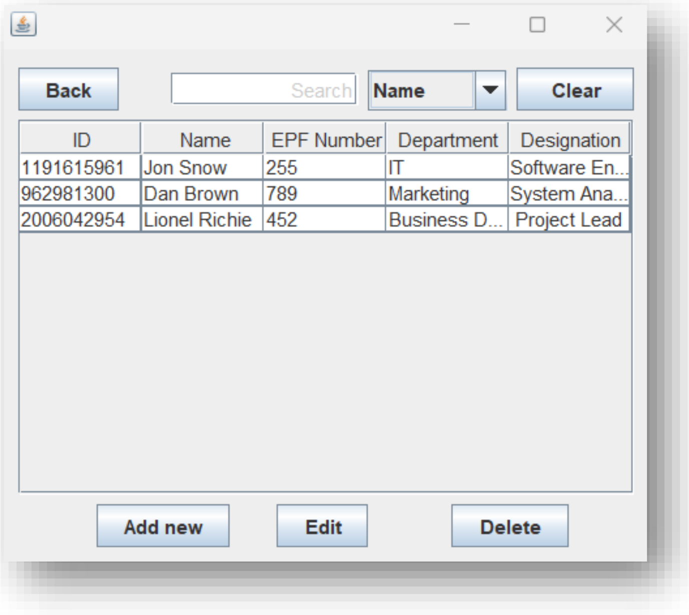
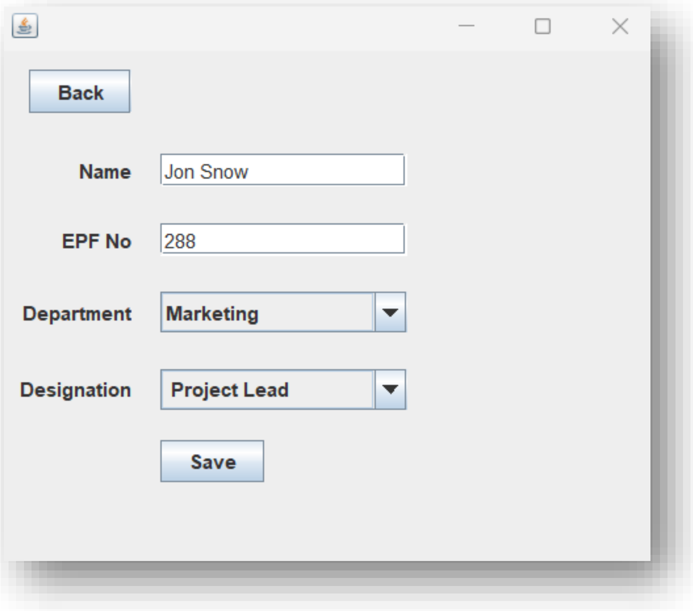
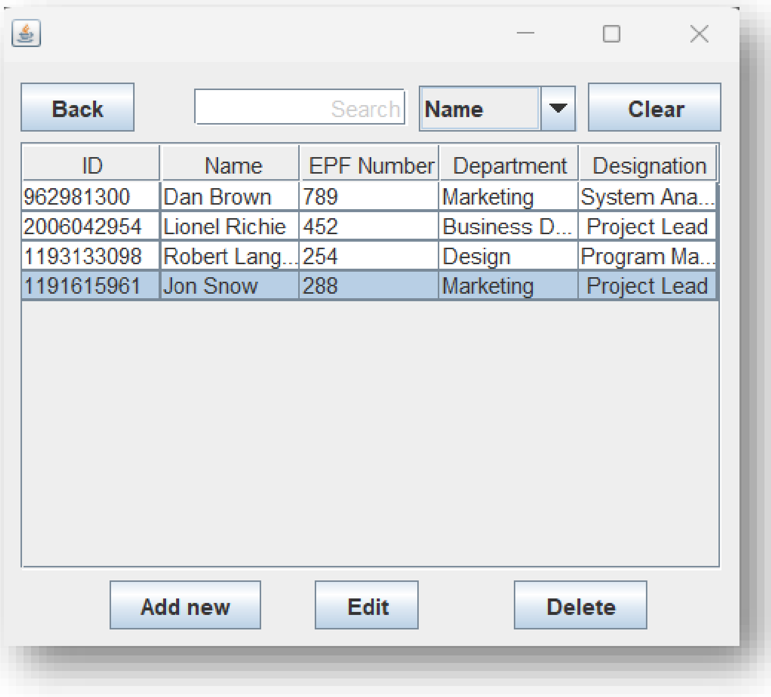
Steps	*Select "Employees" from the menu. *Select "Add new" from the menu. *Put the employee's name and EPF number in. *Choose the appropriate department and title for your designation. *Press the "Add" button.
	 Name: Lionel Richie EPF No: 452 Department: Business Developm... Designation: Project Lead
Result	The addition of the personnel went well.
	 Back Search Name Clear Add new Edit Delete ID, Name, EPF Number, Department, Designation 1191615961, Jon Snow, 255, IT, Software En... 962981300, Dan Brown, 789, Marketing, System Ana... 2006042954, Lionel Richie, 452, Business D..., Project Lead

Table 5:Add Employee

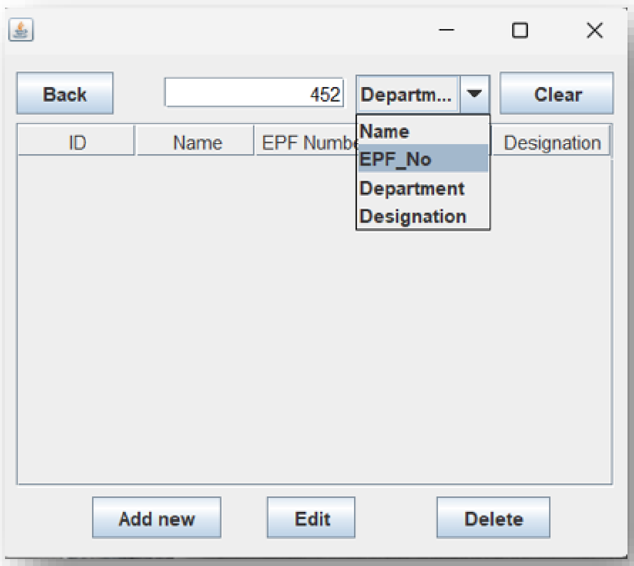
3.5.7 Editing entered employee information

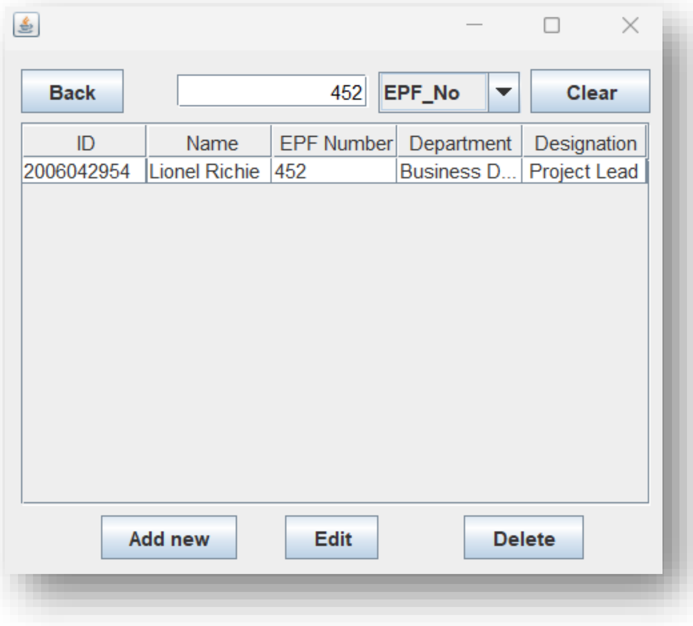
Steps	*To edit, select the column. *Select the "Edit" option. *Revision of pertinent data *Select the "Save" option.
	 Name: Jon Snow EPF No: 288 Department: Marketing Designation: Project Lead
Result	The updated data is displayed accurately.
	

ID	Name	EPF Number	Department	Designation
962981300	Dan Brown	789	Marketing	System Ana...
2006042954	Lionel Richie	452	Business D...	Project Lead
1193133098	Robert Lang...	254	Design	Program Ma...
1191615961	Jon Snow	288	Marketing	Project Lead

Table 6:Editing entered employee information

3.5.8 Employee details are searchable by name, department, EPF number, and other criteria using HR Manager.

Steps	*To search, pick one thing from the list that includes Name, EPF_No., Department, Designation, etc. *copying the personnel information found in the search field associated with that option.
	 Back 452 Departm... Clear ID Name EPF Numb Designation Name EPF_No Department Designation Add new Edit Delete
Result	It is possible to successfully locate employee details by department, title, name, EPF number, etc.

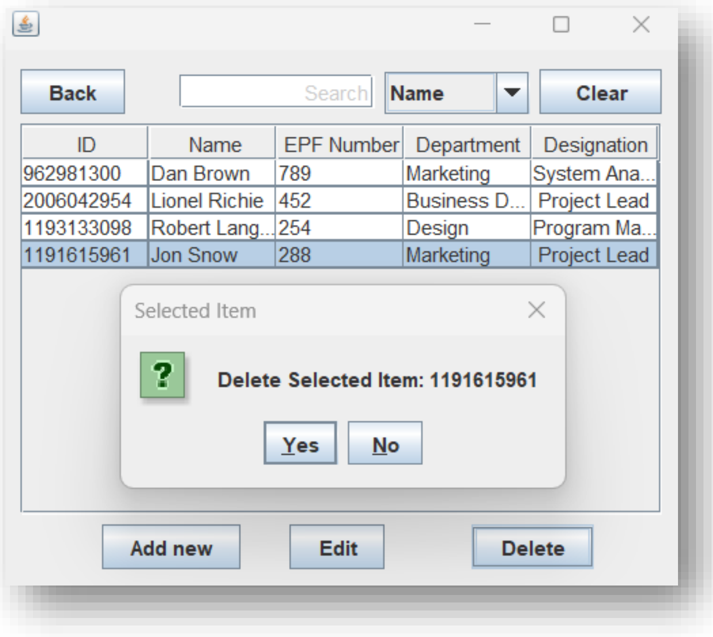
	

Back | 452 | EPF_No | Clear

ID	Name	EPF Number	Department	Designation
2006042954	Lionel Richie	452	Business D...	Project Lead

Table 7:Search Employee Details

3.5.9 Delete the entered employee information.

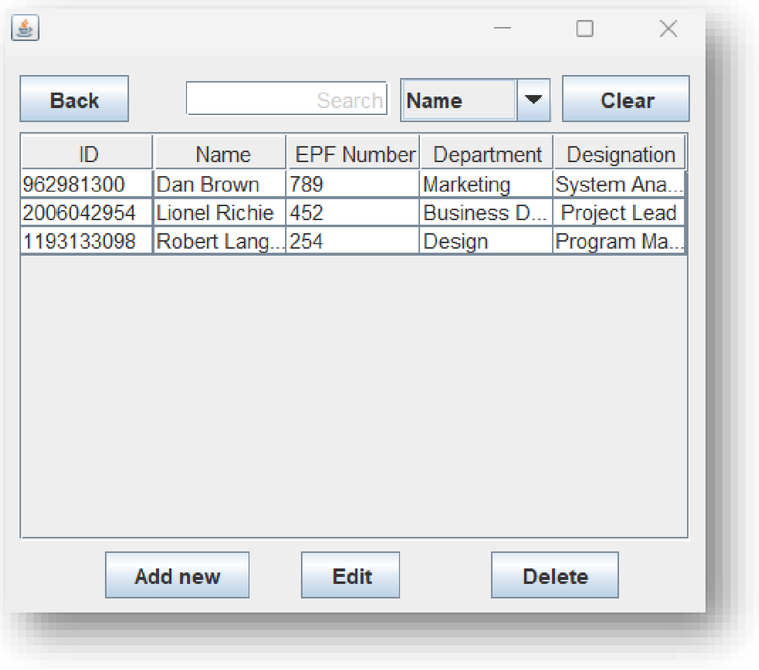
Steps	*Choose the column for employee information that needs to be removed. *Select the "Delete" option. *Choose between the "Yes" and "No" buttons.
	

Back | Search | Name | Clear

ID	Name	EPF Number	Department	Designation
962981300	Dan Brown	789	Marketing	System Ana...
2006042954	Lionel Richie	452	Business D...	Project Lead
1193133098	Robert Lang...	254	Design	Program Ma...
1191615961	Jon Snow	288	Marketing	Project Lead

Selected Item

Delete Selected Item: 1191615961

Result	The employee data was effectively erased.
	

ID	Name	EPF Number	Department	Designation
962981300	Dan Brown	789	Marketing	System Ana...
2006042954	Lionel Richie	452	Business D...	Project Lead
1193133098	Robert Lang...	254	Design	Program Ma...

Table 8:Delete the entered employee information

3.5.10 Creating an account for HR Assistant

Steps	*Select "Users" from the menu. *Select "Add new" from the menu. *Put in the password and user name. *Select "Add User" from the menu.

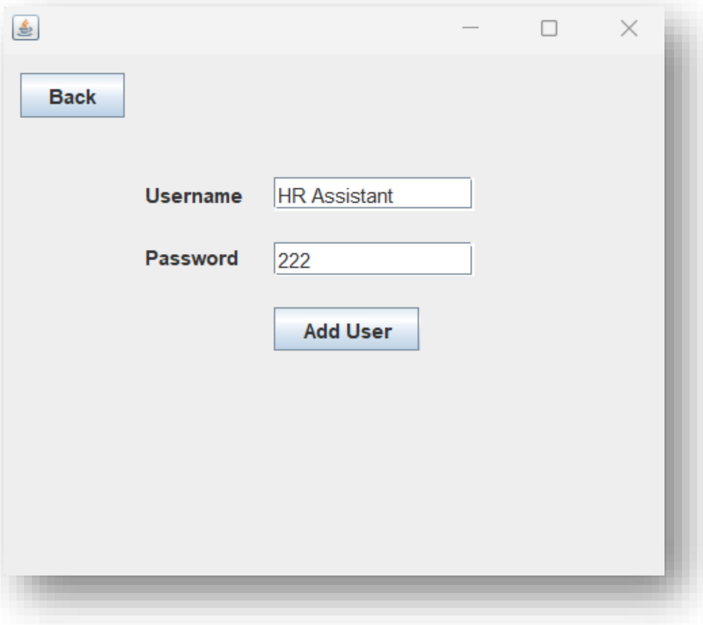
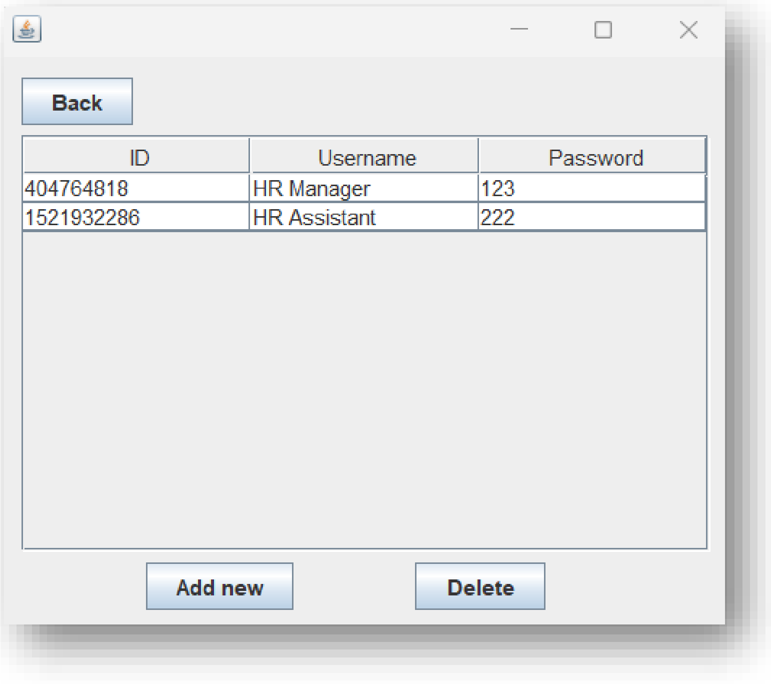
	 Username: HR Assistant Password: 222
Result	The account for the HR Assistant has been successfully created.
	 ID | Username | Password 404764818 | HR Manager | 123 1521932286 | HR Assistant | 222

Table 9:Creating an account for HR Assistant

3.5.11 Logging to HR Assistant Account

3.5.11.1 Enter the correct password.

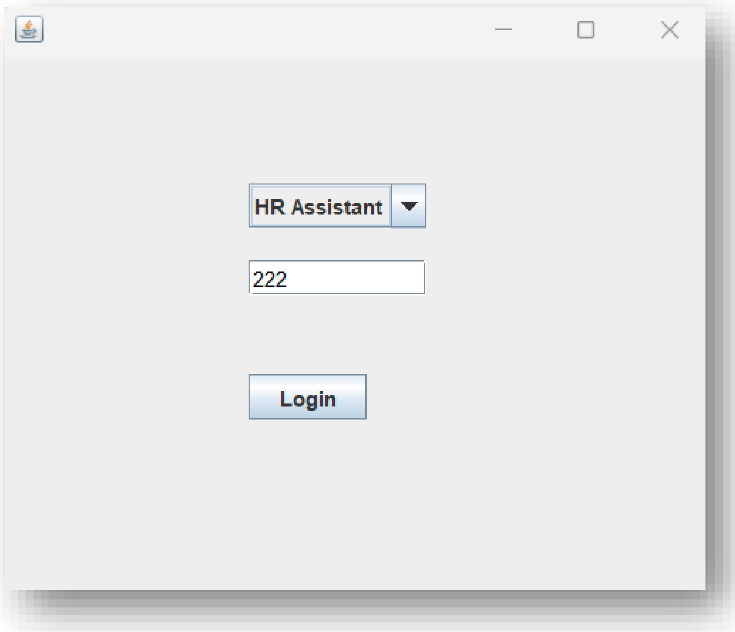
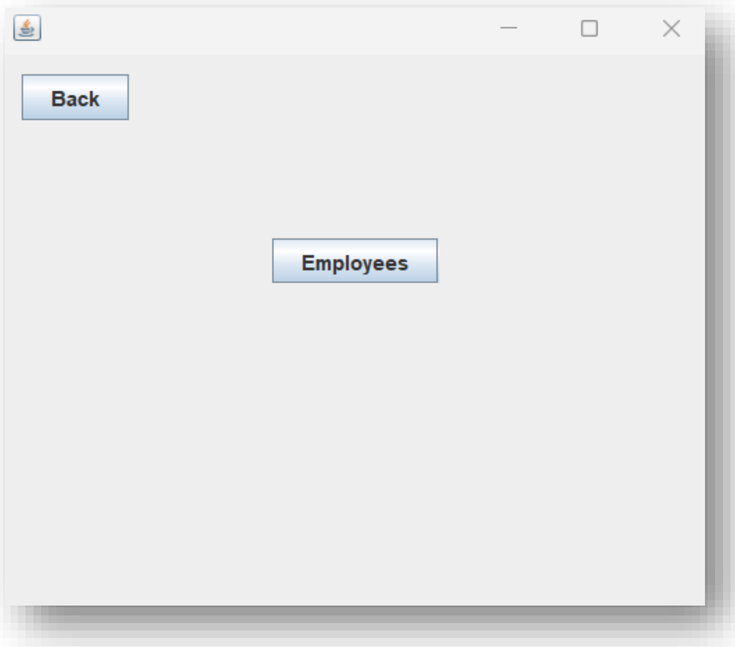
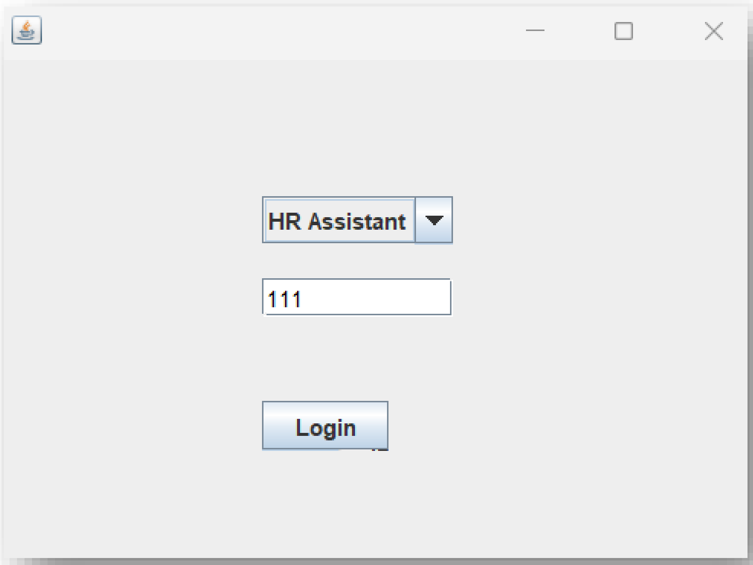
Steps	*Choose the accurate user name. *Kindly input the accurate password. *Select the "Login" option. ▪
	
Result	Entering the HR Assistant account successfully.
	

Table 10:Enter the correct password

3.5.11.2 Enter the incorrect password.

Steps	*Make sure you choose the right user name. *Put in the wrong password. *Select "Login" from the menu.
	
Result	An error notice shows up.

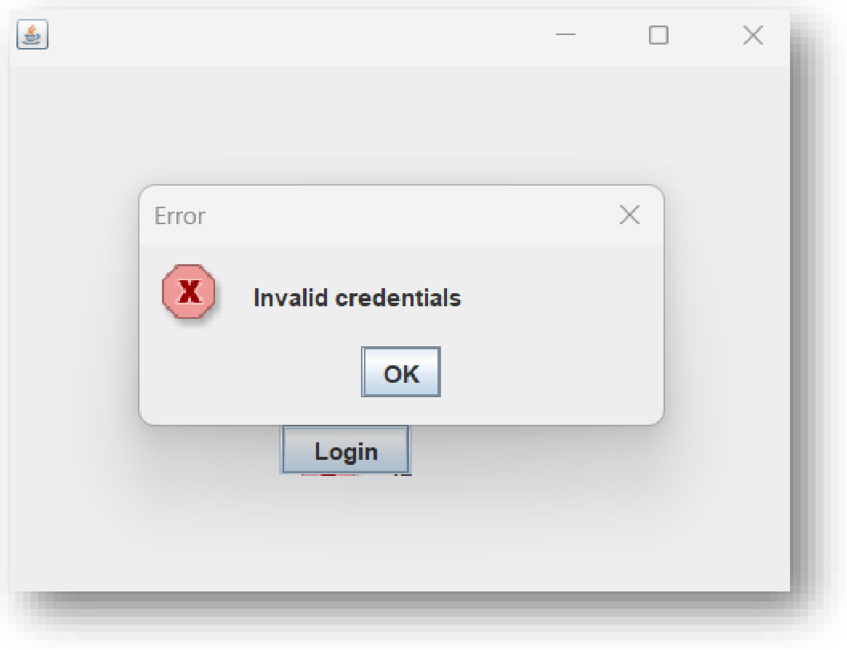
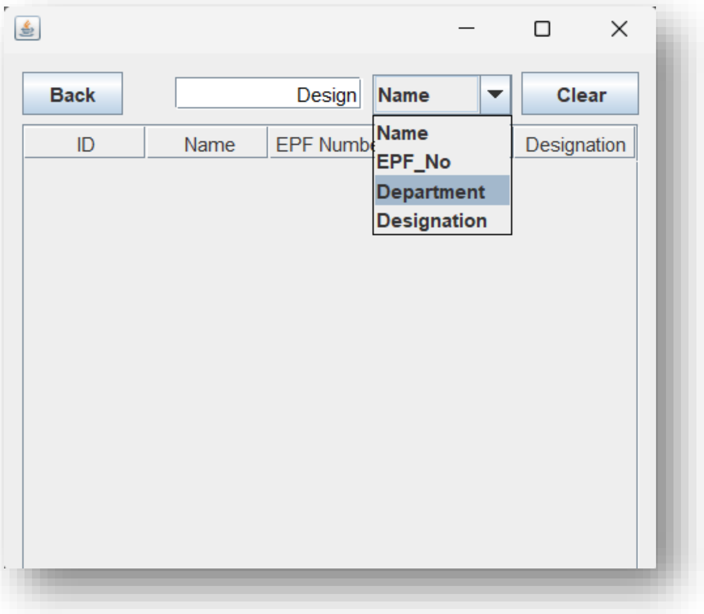
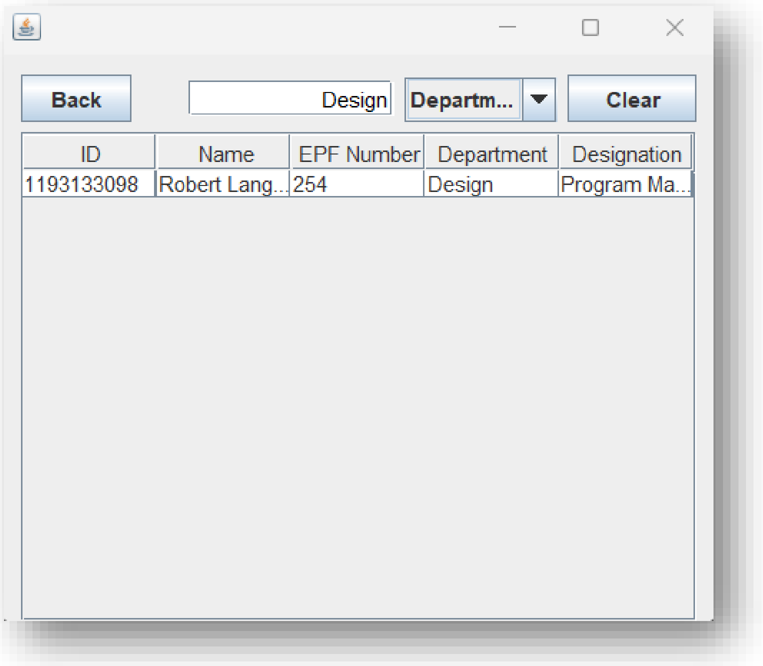
	 Error Invalid credentials OK Login

Table 11:Enter the incorrect password

3.5.11.3 HR Assistant employee details search based on department, designation, name, EPF number, etc.

Steps	*Select "Employees" from the menu. *To search, pick one thing from the list that includes Name, EPF_No., Department, Designation, etc. *copying the personnel information found in the search field associated with that option.

	 Back Design Name Clear ID Name EPF Numbe Designation Name EPF_No Department Designation
Result	It is possible to successfully locate employee details by department, title, name, EPF number, etc.
	 ID: 1193133098; Name: Robert Lang...; EPF Number: 254; Department: Design; Designation: Program Ma...

3.6. Data Saving and Retrieval:

Employee information, departments, and designations are automatically saved by the system to a file for later use.

To guarantee system continuity, the system loads the previously saved data when it is launched.

3.7. Troubleshooting:

Should you experience any problems while utilising the system, please get in touch with the support staff for help.

3.8. Feedback:

We appreciate your opinions! Please contact the support team to report any problems or recommendations you may have while using the system.

3.9 Conclusion:

We are grateful that you are utilising the HR Management System designed for the "Connect Me" mobile service provider. We hope that this method increases overall efficiency and streamlines your employee management procedure. Do not hesitate to contact us if you need further help or if you have any questions.

References

[1] "Object-Oriented Programming (OOP) Concepts", GeeksforGeeks, Available online: https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/

[2] "Java File Handling - Serialization", Tutorialspoint, Available online: https://www.tutorialspoint.com/java/java_serialization.htm

[3] "Introduction to UML", Sparx Systems, Available online: https://www.sparxsystems.com/resources/uml2_tutorial/uml2_use_case.html

[4] "Java GUI Tutorial - JFrame, JPanel, Layouts, Event Handling, and Listeners", Available online: https://www.javatpoint.com/java-jframe

[5] "Java File Handling - Reading and Writing Files", Tutorialspoint, Available online: https://www.tutorialspoint.com/java/java_files_io.htm

[6] "Java Swing Tutorial - JList", Available online: https://www.javatpoint.com/java-jlist

[7] "Java Abstract Class and Methods", GeeksforGeeks, Available online: https://www.geeksforgeeks.org/abstract-classes-in-java/

[8] "Java Inheritance", W3Schools, Available online: https://www.w3schools.com/java/java_inheritance.asp

[9] "Java Encapsulation", Tutorialspoint, Available online: https://www.tutorialspoint.com/java/java_encapsulation.htm

[10] "Java Polymorphism", Oracle Documentation, Available online: https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html

[11] "Introduction to Java GUI", Oracle Documentation, Available online: https://docs.oracle.com/javase/tutorial/uiswing/start/about.html