EMPLOYEE MANAGEMENT SYSTEM

A MINI-PROJECT BY:

Manasa Visakai.S 230701172

Harine.AS 230701516

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2023

BONAFIDE CERTIFICATE

Certified that this project **"EMPLOYEE MANAGEMENT SYSTEM"** is the bonafide work of **"HARINE. AS , MANASA VISAKAI.S"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

Mrs. K.Maheshmeena
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonom ous),
Thandalam,Chennai-602 105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The Employee Management System is a Java-based software application created to make managing employee records easier and more efficient. Built with JavaFX for its user-friendly interface and SQLite for lightweight and reliable data storage, this project focuses on streamlining essential tasks like adding, updating, and deleting employee records. The development was carried out in IntelliJ IDEA, ensuring a robust and efficient coding environment.

The system features a visually appealing and intuitive interface. With colorful tables, bold headings, and an organized layout, navigating through the dashboard feels natural and engaging. SQLite, as the database backend, keeps things simple yet effective by offering reliable data storage and retrieval without the need for complex server setups. This combination ensures that the system is both easy to use and deploy.

This project was born out of a need to replace traditional, manual methods of employee management, which can often be time-consuming and error-prone. By automating these processes, the system reduces human errors, saves time, and provides instant access to well-organized data. Its modular design also makes it adaptable for future enhancements, such as adding analytics or role-based access..

Although it started as a mini-project, this Employee Management System demonstrates how technology can solve real-world problems in practical and meaningful ways. It highlights the importance of automation in improving organizational efficiency and serves as a strong foundation for building more advanced systems in the future.

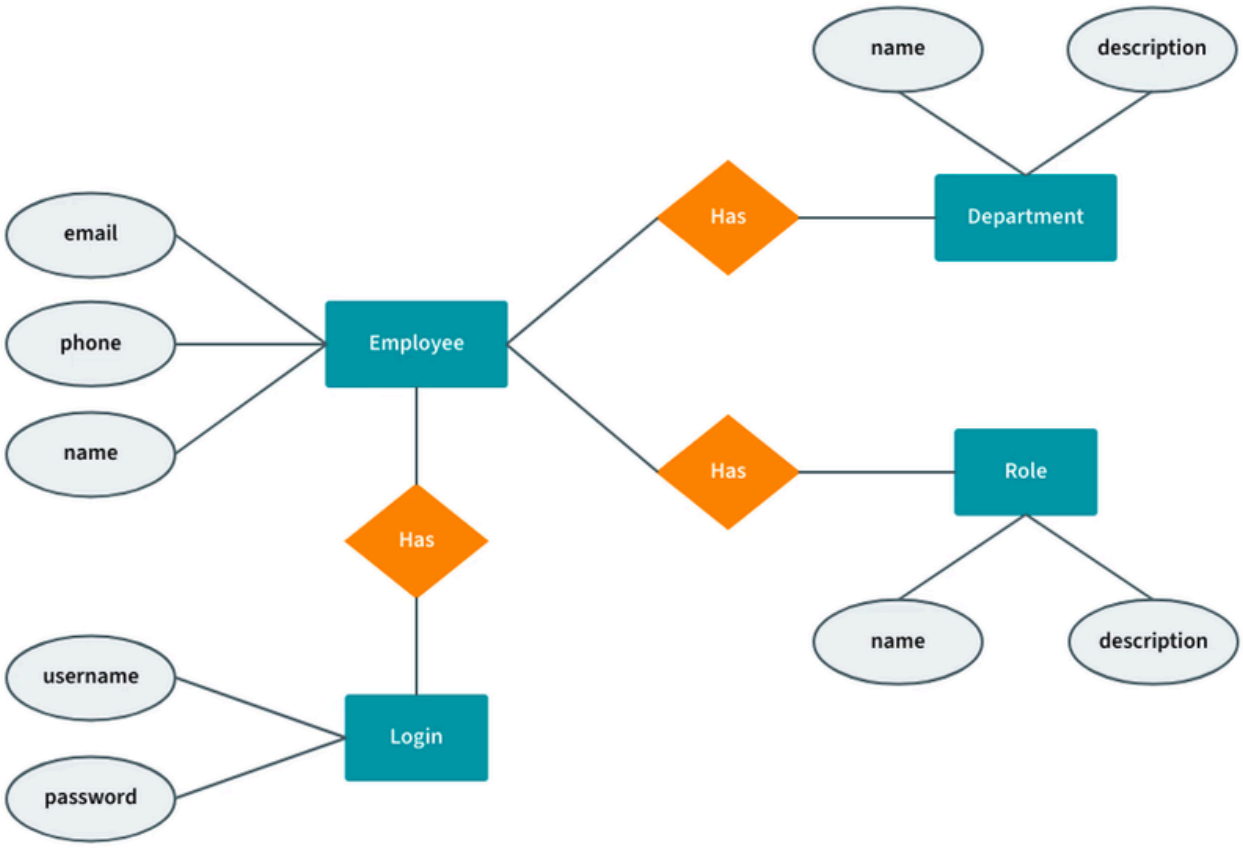
TABLE OF CONTENTS

CRUDE OPERATIONS

CRUD Operations in the Employee Management System

1. Create (Insert Employee): Adds a new employee record to the SQLite database using the INSERT query.
2. Read (View Employees): Retrieves and displays all employee records using the SELECT query.
3. Update (Update Employee): Modifies an existing employee's details using the UPDATE query based on their ID.
4. Delete (Remove Employee): Deletes an employee record from the database using the DELETE query by matching the ID.

ER DIAGRAM



INTRODUCTION

1.1 INTRODUCTION

The project streamlines employee management by providing a digital platform for organizations to handle employee records efficiently. It allows administrators to manage operations such as adding, updating, and deleting employee details, ensuring organized and error-free data management. This system improves accessibility, reduces manual effort, and promotes automation.

1.2 IMPLEMENTATION

The Employee Management System discussed here is implemented using JavaFX for the graphical user interface and SQLite as the backend database. The project was developed using IntelliJ IDEA as the integrated development environment, ensuring a seamless coding and testing process.

1.3 SCOPE OF THE PROJECT

The system is designed to provide a professional and user-friendly experience for managing employee data. It enables administrators to store all employee records in one place, ensuring data security and consistency. The organized approach saves time and effort, making the process efficient for both small and large organizations. The modular design of the system allows for future scalability and the integration of additional features.

1.4 WEBSITE FEATURES

- Registration and login pages for secure access.
- A custom dashboard for administrators.
- Table display of employee data with options for CRUD operations (Create, Read, Update, Delete).
- Interactive GUI with a vibrant color theme and user-friendly navigation.
- Integration with SQLite for efficient and reliable data storage.

SYSTEM SPECIFICATION

2.1 HARDWARE SPECIFICATIONS

- Processor: Intel i5 (or equivalent)
- Memory Size: 4GB (minimum)
- Hard Disk: 500GB of free space

2.2 SOFTWARE SPECIFICATIONS

- Programming Language: Java
- Front-End: JavaFX
- Back-End: SQLite
- Operating System: Windows 10

SAMPLE CODE

3.1 HOME PAGE DESIGN

```
@Override
public void start(Stage primaryStage) {
   primaryStage.setTitle("Employee Management System");

   BorderPane mainLayout = new BorderPane();

   Label heading = new Label("Employee Management System");
        heading.setStyle("-fx-font-size: 20px; -fx-font-weight: bold; -fx-border-color: black; -fx-background-color: lightblue; -fx-padding: 10px;");
   heading.setMaxWidth(Double.MAX_VALUE);
   heading.setAlignment(javafx.geometry.Pos.CENTER);
   mainLayout.setTop(heading);

   VBox buttonLayout = new VBox(10);
   buttonLayout.setStyle("-fx-alignment: center; -fx-padding: 20px;");

   Button addButton = new Button("Add Employee");
   Button updateButton = new Button("Update Employee");
   Button deleteButton = new Button("Delete Employee");
   Button viewButton = new Button("View Employees");

        buttonLayout.getChildren().addAll(addButton, updateButton, deleteButton, viewButton);
   mainLayout.setCenter(buttonLayout);

   Scene mainScene = new Scene(mainLayout, 500, 400);
   primaryStage.setScene(mainScene);
   primaryStage.show();
}
```

3.2 REGISTRATION PAGE DESIGN

```
private void openAddEmployee() {
    Stage addStage = new Stage();
    addStage.setTitle("Add Employee");

    VBox layout = new VBox(10);
    layout.setStyle("-fx-padding: 20px;");

    Label empIdLabel = new Label("Employee ID:");
    TextField empIdField = new TextField();

    Label empNameLabel = new Label("Employee Name:");
    TextField empNameField = new TextField();

    Label empRoleLabel = new Label("Employee Role:");
    TextField empRoleField = new TextField();

    Button addButton = new Button("Add");
    addButton.setOnAction(e -> {
        String empId = empIdField.getText();
        String empName = empNameField.getText();
        String empRole = empRoleField.getText();

        if (!empId.isEmpty() && !empName.isEmpty() && !empRole.isEmpty()) {
            employeeData.add(new Employee(empId, empName, empRole));
            Alert successAlert = new Alert(Alert.AlertType.INFORMATION, "Employee added
successfully!");
            successAlert.showAndWait();
            addStage.close();
        } else {
            Alert errorAlert = new Alert(Alert.AlertType.ERROR, "All fields are required!");
            errorAlert.showAndWait();
        }
    });

    layout.getChildren().addAll(empIdLabel, empIdField, empNameLabel, empNameField,
empRoleLabel, empRoleField, addButton);
    Scene addScene = new Scene(layout, 300, 300);
    addStage.setScene(addScene);
    addStage.show();
}
```

3.3 LOGIN PAGE DESIGN

```
Label usernameLabel = new Label("Username:");
TextField usernameField = new TextField();

Label passwordLabel = new Label("Password:");
PasswordField passwordField = new PasswordField();

Button loginButton = new Button("Login");
loginButton.setOnAction(e -> {
    String username = usernameField.getText();
    String password = passwordField.getText();
    if (username.equals("admin") && password.equals("password")) {
        Alert success = new Alert(Alert.AlertType.INFORMATION, "Login Successful!");
        success.showAndWait();
    } else {
        Alert failure = new Alert(Alert.AlertType.ERROR, "Invalid Credentials!");
        failure.showAndWait();
    }
});
```

3.4 DASHBOARD DESIGN

```
VBox buttonLayout = new VBox(10);
buttonLayout.setStyle("-fx-alignment: center; -fx-padding: 20px;");

Button addButton = new Button("Add Employee");
Button updateButton = new Button("Update Employee");
Button deleteButton = new Button("Delete Employee");
Button viewButton = new Button("View Employees");

addButton.setOnAction(e -> openAddEmployee());
updateButton.setOnAction(e -> openUpdateEmployee());
deleteButton.setOnAction(e -> openDeleteEmployee());
viewButton.setOnAction(e -> openViewEmployees());

buttonLayout.getChildren().addAll(addButton,updateButton,  deleteButton,
viewButton);
mainLayout.setCenter(buttonLayout);
```

3.4 DASHBOARD DESIGN

```
VBox buttonLayout = new VBox(10);
buttonLayout.setStyle("-fx-alignment: center; -fx-padding: 20px;");

Button addButton = new Button("Add Employee");
Button updateButton = new Button("Update Employee");
Button deleteButton = new Button("Delete Employee");
Button viewButton = new Button("View Employees");

addButton.setOnAction(e -> openAddEmployee());
updateButton.setOnAction(e -> openUpdateEmployee());
deleteButton.setOnAction(e -> openDeleteEmployee());
viewButton.setOnAction(e -> openViewEmployees());

buttonLayout.getChildren().addAll(addButton,updateButton,  deleteButton,
viewButton);
mainLayout.setCenter(buttonLayout);
```

3.5 ADDING AN EMPLOYEE BACKEND

```
public static String addEmployee(String id, String name, String role) {
    employees.add(new Employee(id, name, role));
    return "Employee added successfully!";
}
```

3.6 UPDATING AN EMPLOYEE BACKEND

```
public static String updateEmployee(String id, String name, String role) {
    for (Employee emp : employees) {
        if (emp.getId().equals(id)) {
            emp.setName(name);
            emp.setRole(role);
            return "Employee updated successfully!";
        }
    }
    return "Employee not found!";
}
```

3.7 DELETING AN EMPLOYEE BACKEND

```
public static String deleteEmployee(String id) {
    for (Employee emp : employees) {
        if (emp.getId().equals(id)) {
            employees.remove(emp);
            return "Employee deleted successfully!";
        }
    }
    return "Employee not found!";
}
```

3.8 VIEWING EMPLOYEES BACKEND

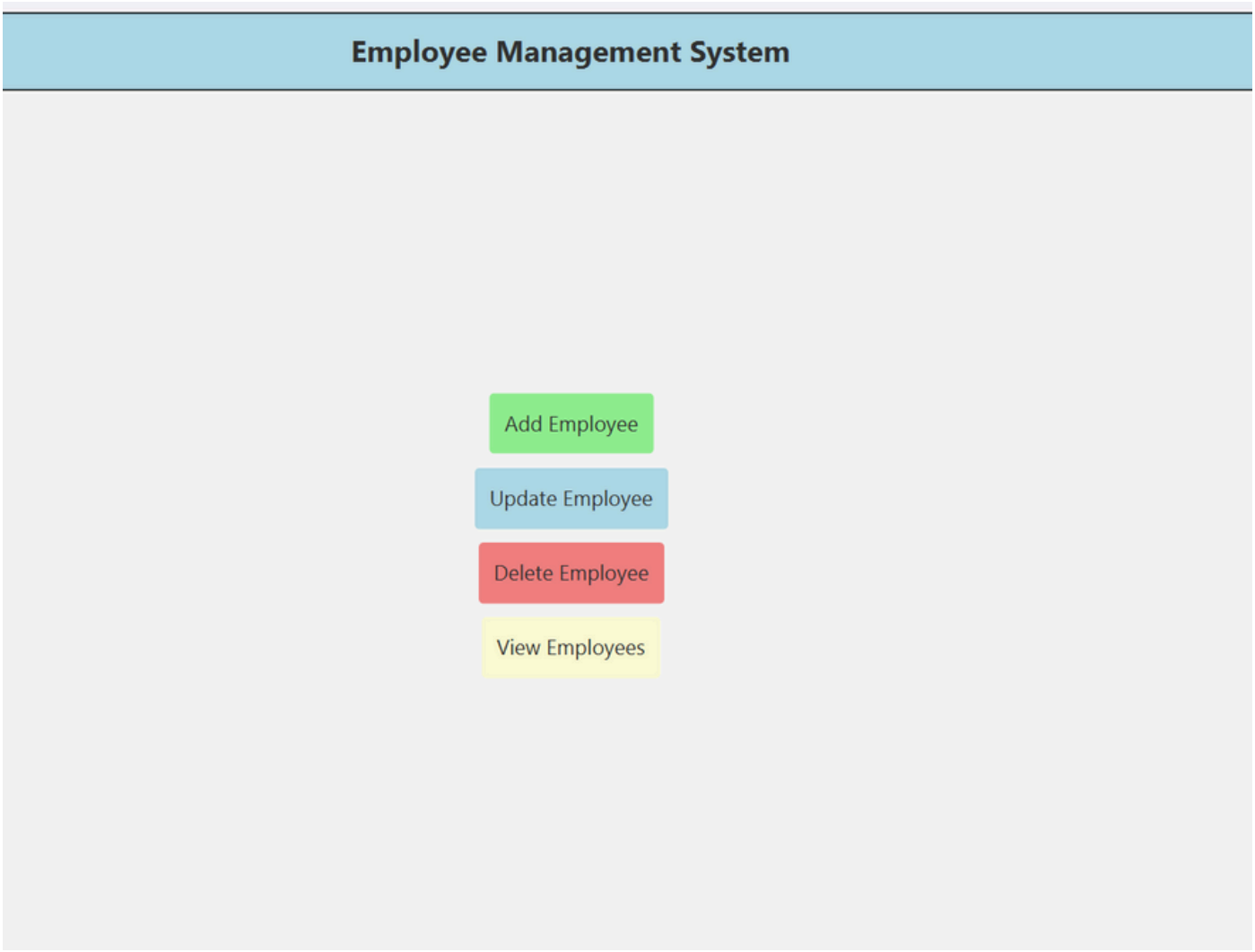
```
public static List<Employee> getAllEmployees() {
    return employees;
}
```

3.9 EMPLOYEE CLASS DESIGN

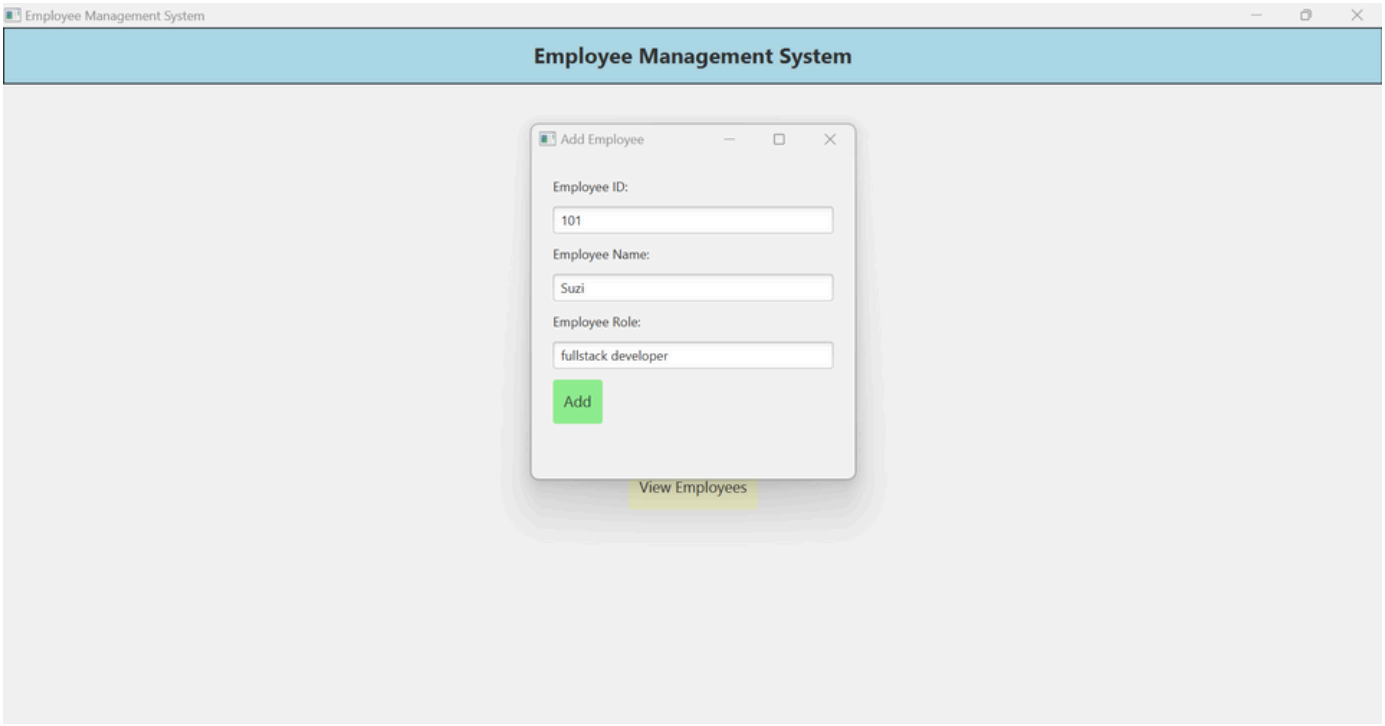
```
public class Employee {
    private SimpleStringProperty id;
    private SimpleStringProperty name;
    private SimpleStringProperty role;

    public Employee(String id, String name, String role) {
        this.id = new SimpleStringProperty(id);
        this.name = new SimpleStringProperty(name);
        this.role = new SimpleStringProperty(role);
    }

    public String getId() {
        return id.get();
    }

    public String getName() {
        return name.get();
    }

    public String getRole() {
        return role.get();
    }

    public SimpleStringProperty idProperty() {
        return id;
    }

    public SimpleStringProperty nameProperty() {
        return name;
    }

    public SimpleStringProperty roleProperty() {
        return role;
    }
}
```

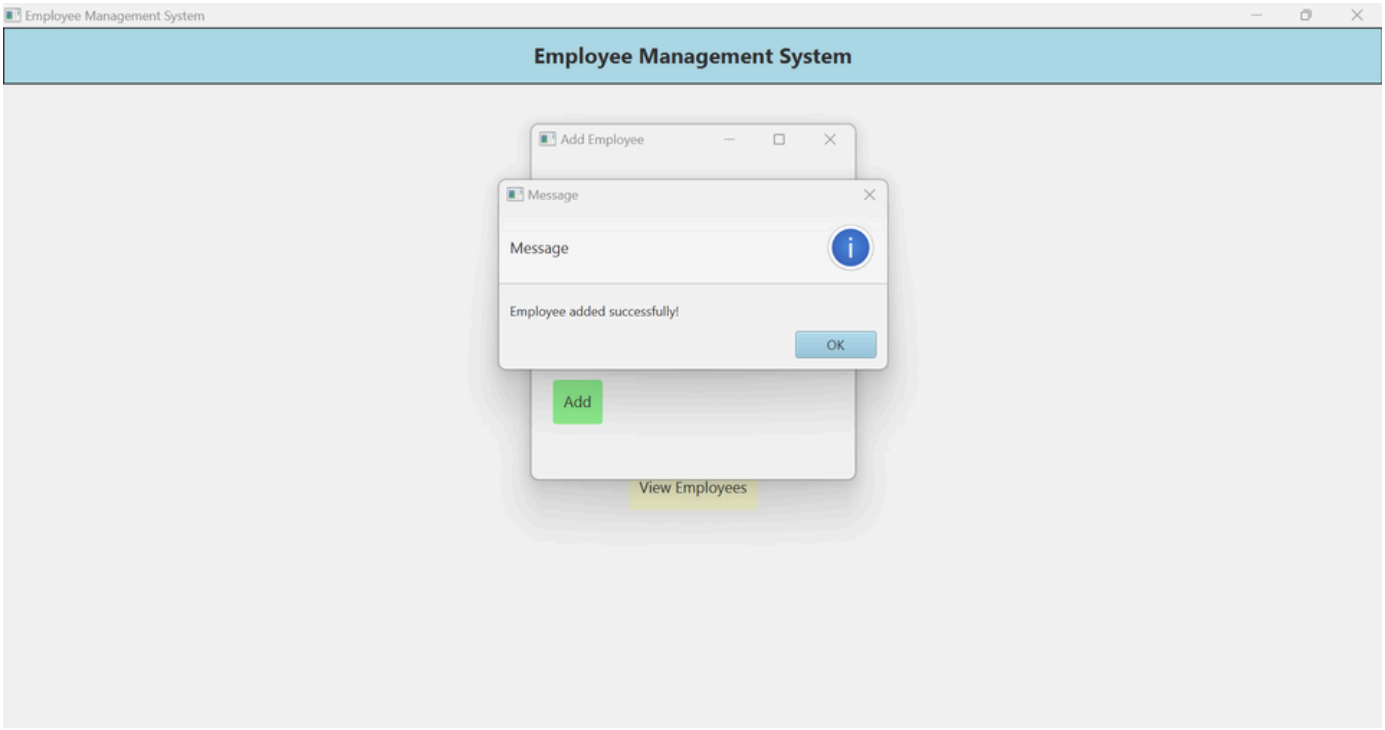
SNAPSHOTS

HOME PAGE

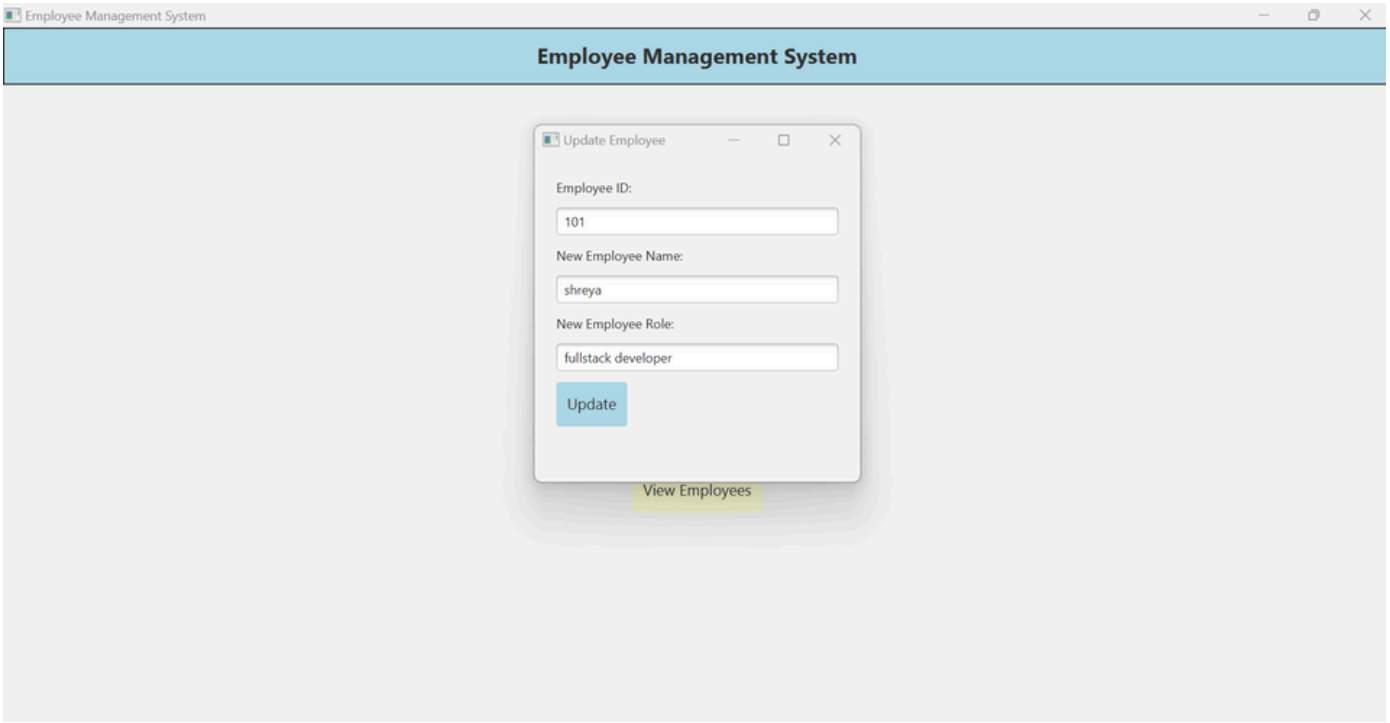


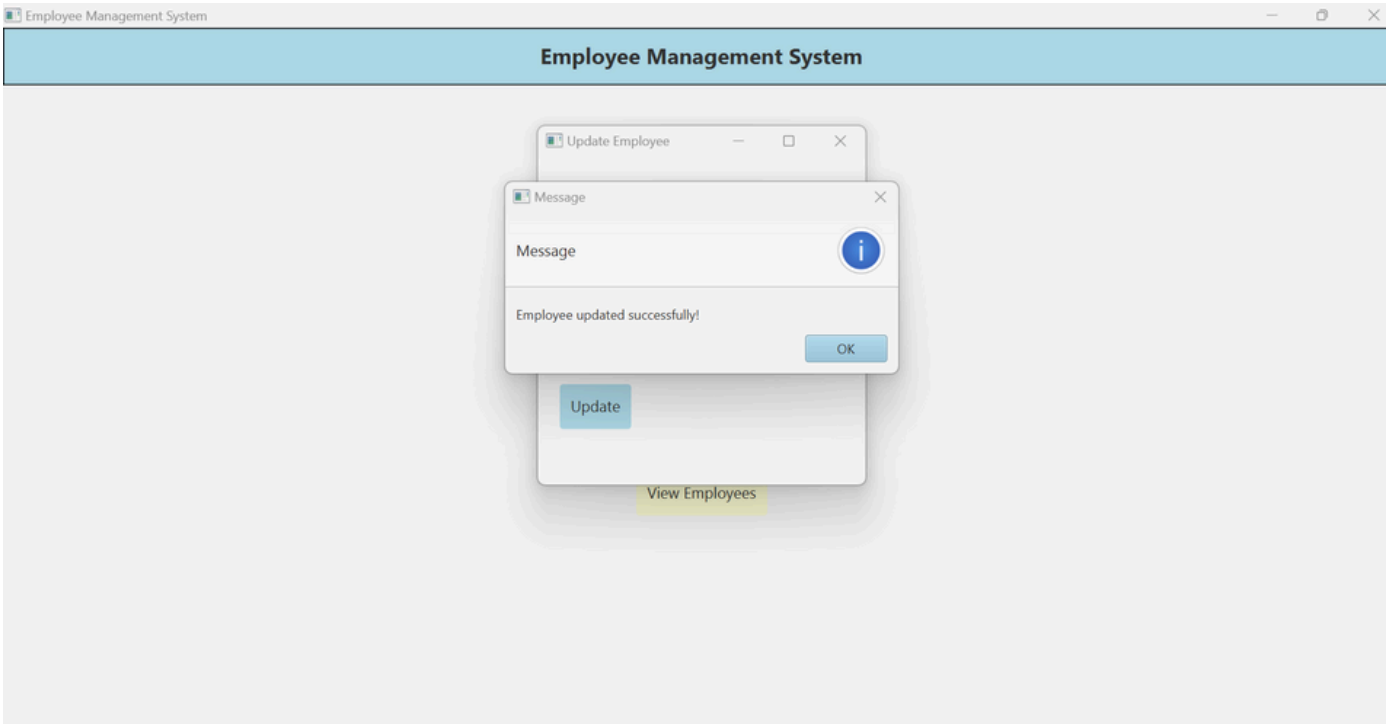
ADD PAGE


Employee Management System
Employee Management System
Add Employee
Employee ID:
101
Employee Name:
Suzi
Employee Role:
fullstack developer
Add
View Employees

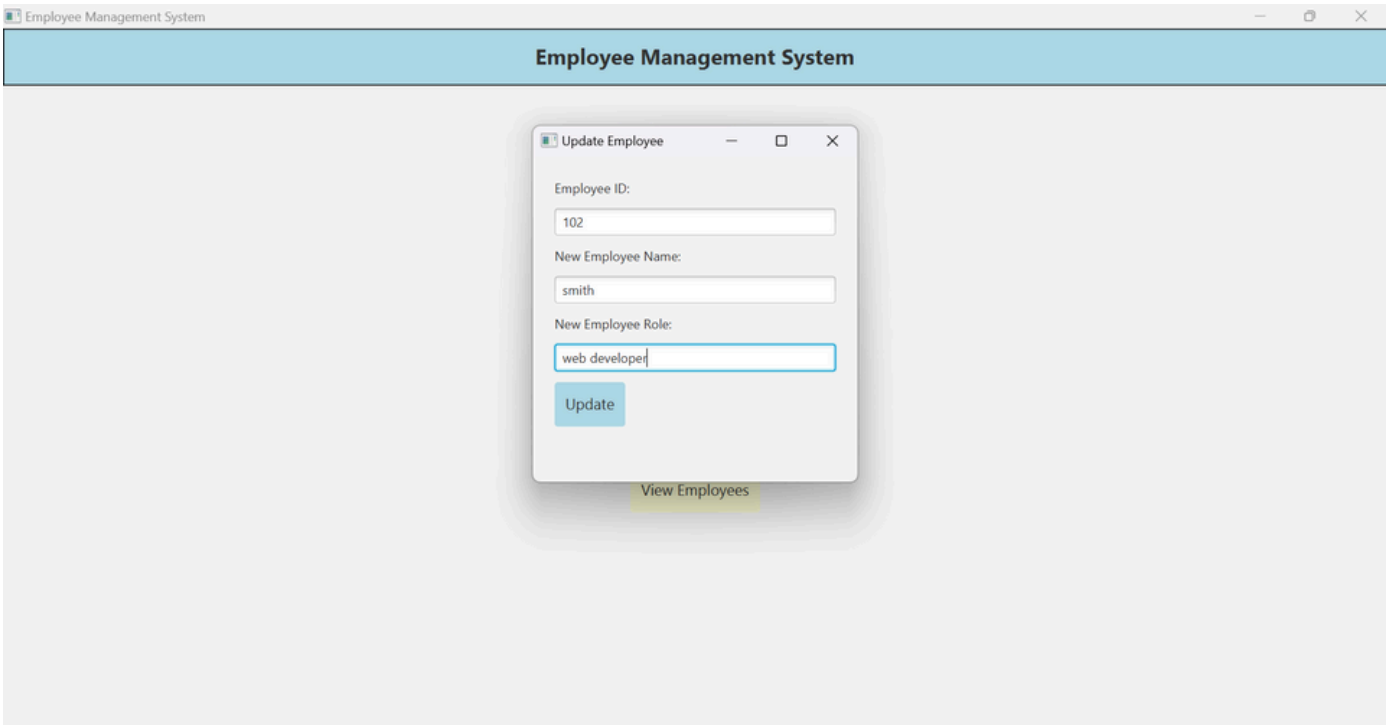

Employee Management System
Employee Management System
Add Employee
Message
Message
Employee added successfully!
OK
Add
View Employees

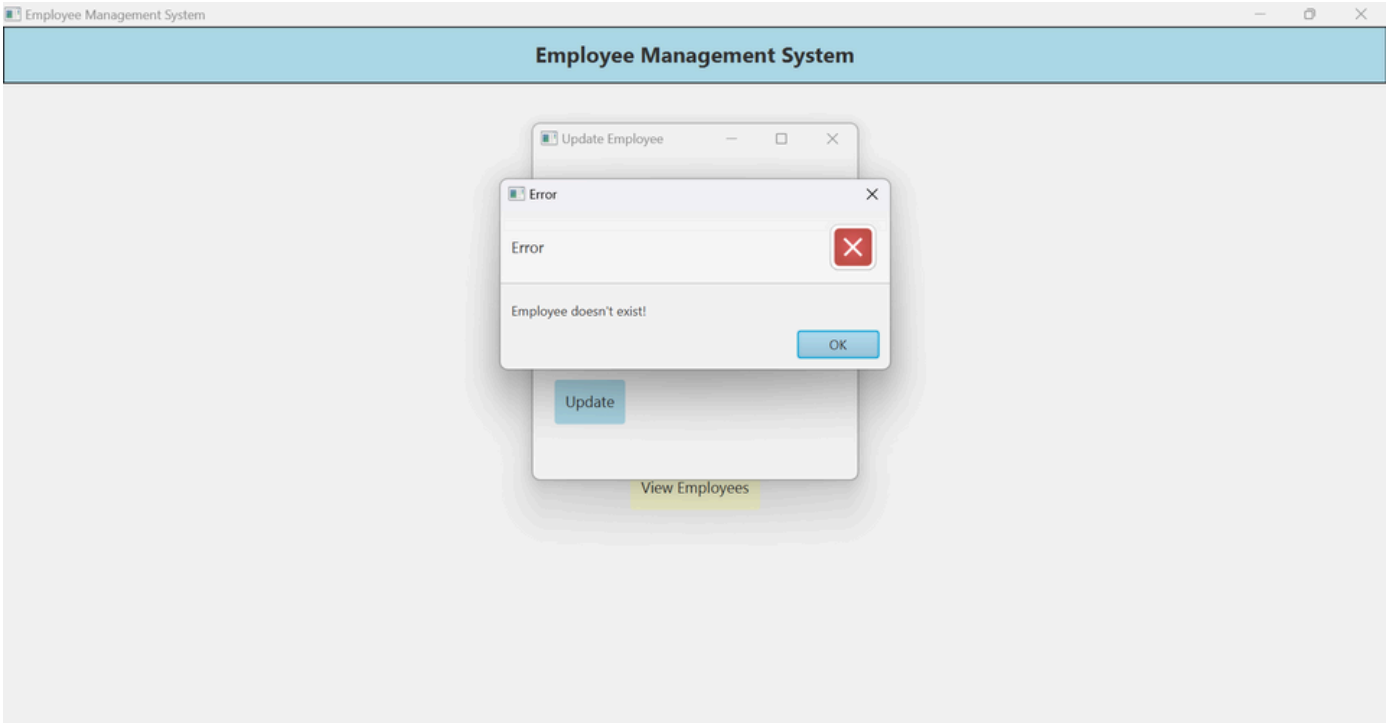
UPDATE PAGE


Employee Management System
Employee Management System
Update Employee
Employee ID:
101
New Employee Name:
shreya
New Employee Role:
fullstack developer
Update
View Employees

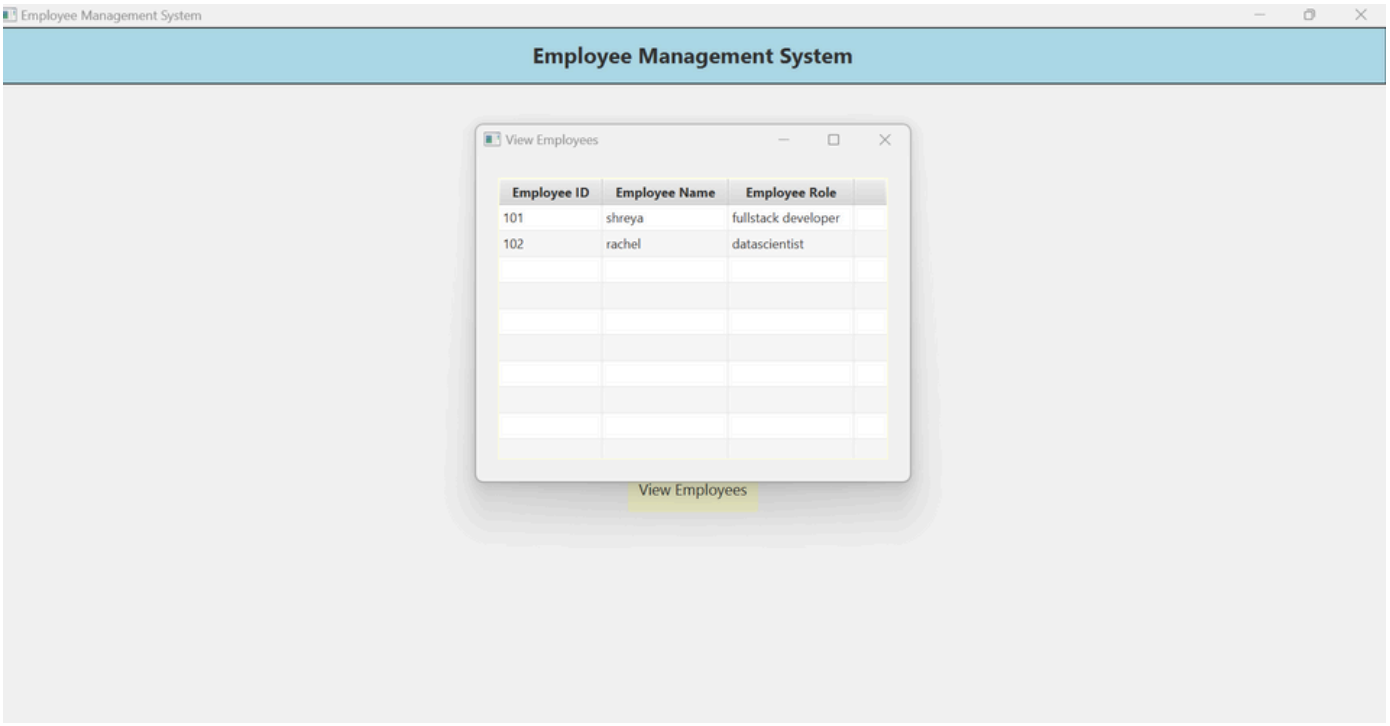

Employee Management System
Employee Management System
Update Employee
Message
Message
Employee updated successfully!
OK
Update
View Employees

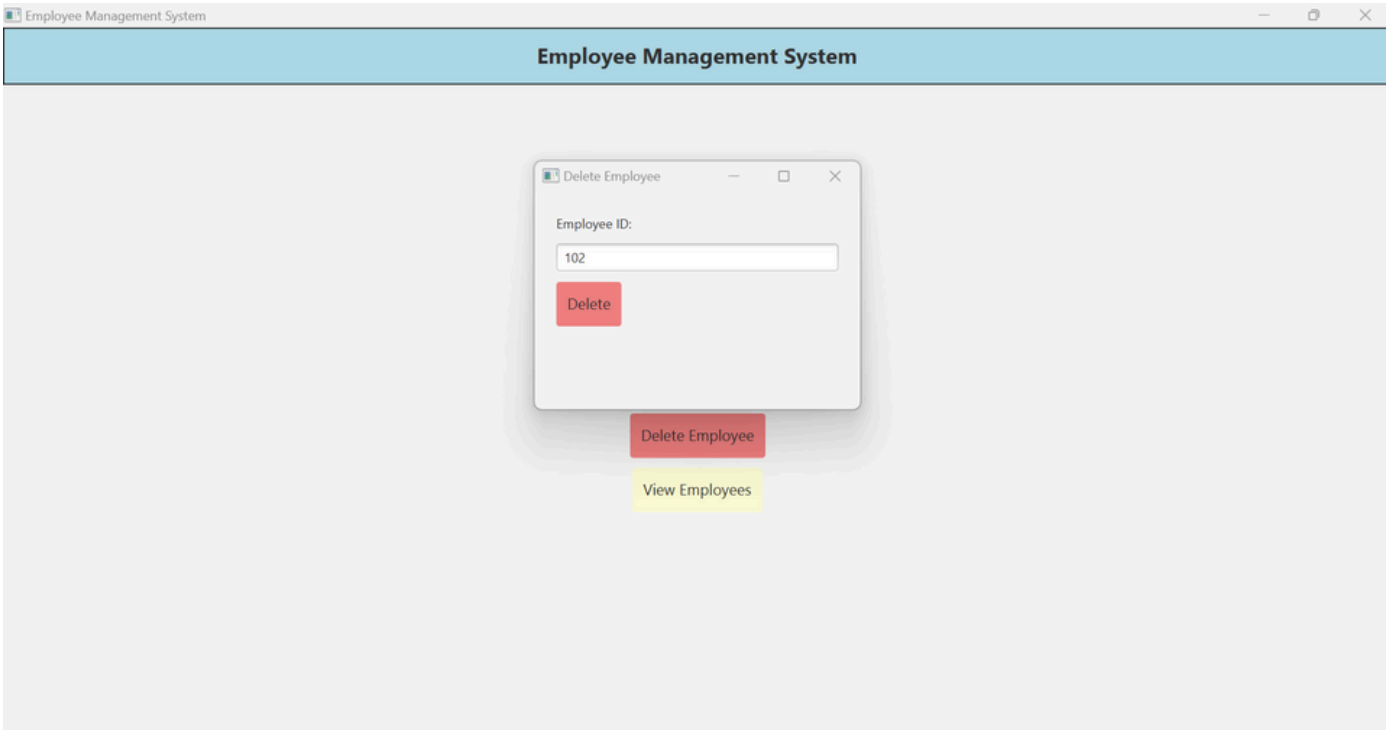
UPDATE PAGE


Employee Management System
Employee Management System
Update Employee
Employee ID:
102
New Employee Name:
smith
New Employee Role:
web developer
Update
View Employees

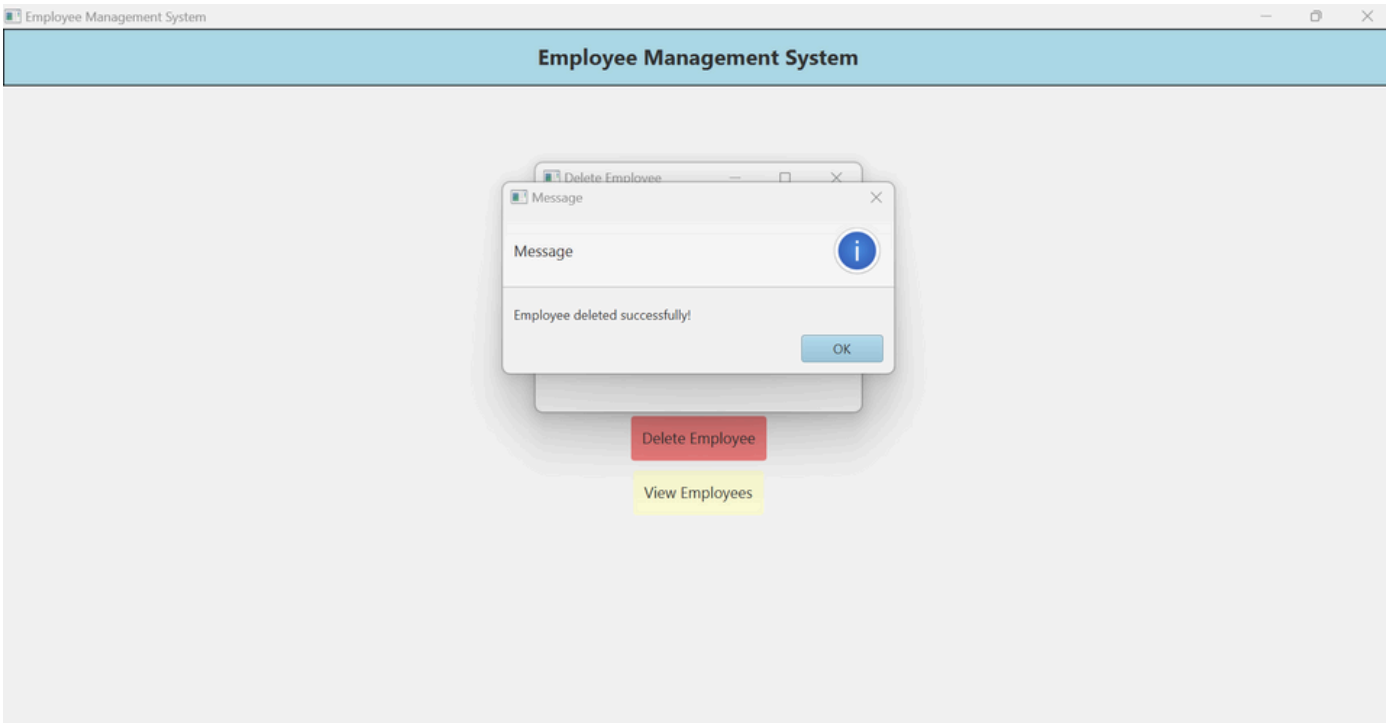

Employee Management System
Employee Management System
Update Employee
Error
Error
Employee doesn't exist!
OK
Update
View Employees

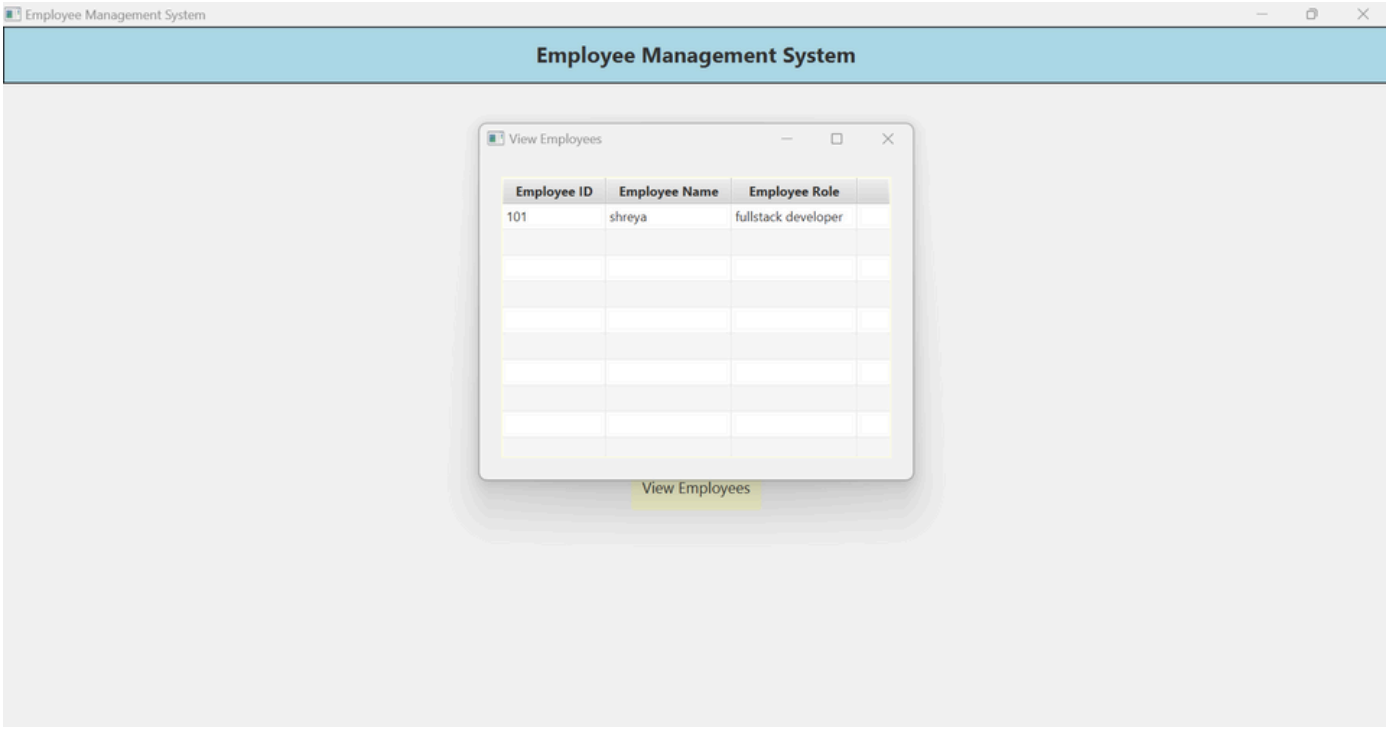
DELETE PAGE


Employee Management System
Employee Management System
View Employees
Employee ID
Employee Name
Employee Role
101
shreya
fullstack developer
102
rachel
datascientist
View Employees

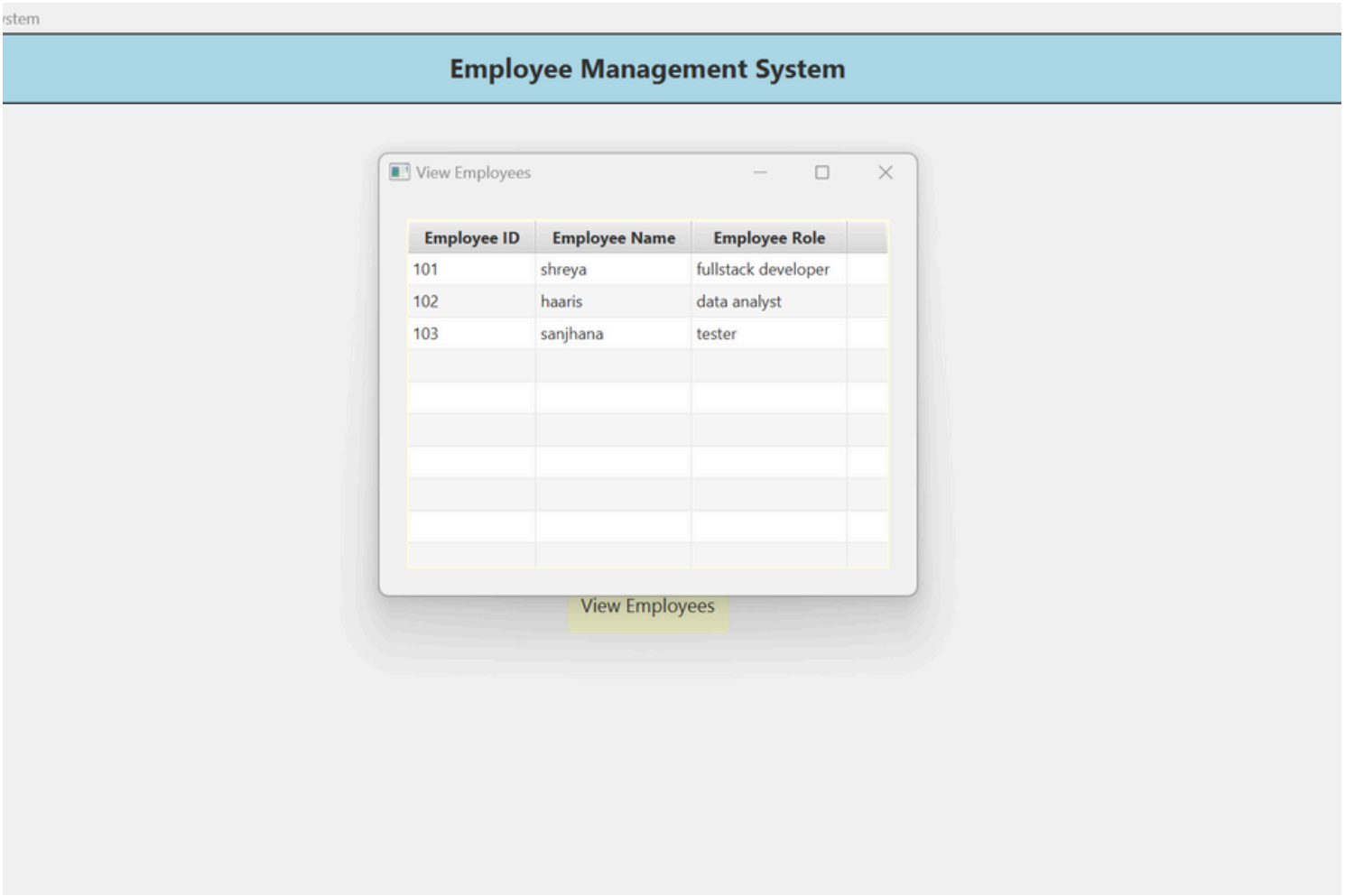

Employee Management System
Employee Management System
Delete Employee
Employee ID:
102
Delete
Delete Employee
View Employees

DELETE PAGE


Employee Management System
Employee Management System
Delete Employee
Message
Message
Employee deleted successfully!
OK
Delete Employee
View Employees


Employee Management System
Employee Management System
View Employees
Employee ID
Employee Name
Employee Role
101
shreya
fullstack developer
View Employees

VIIEW PAGE


Employee Management System
View Employees
Employee ID
Employee Name
Employee Role
101
shreya
fullstack developer
102
haaris
data analyst
103
sanjhana
tester
View Employees

CONCLUSION

The Employee Management System project demonstrates the integration of JavaFX for creating a user-friendly graphical interface with a structured backend for employee operations. By employing concepts like JavaFX bindings, modularity through classes, and an organized workflow, this project not only simplifies employee management but also provides a scalable framework for similar applications. It highlights the importance of clean code practices, reusability, and a seamless user experience. As JavaFX is a versatile framework, the project can be extended further to include database integration, advanced search functionality, and role-based access for a complete professional application.

REFERENCES

1. JavaFX Official Documentation
2. JavaFX Tutorials - CodeGym
3. Introduction to JavaFX | Baeldung
4. JavaFX TableView Documentation
5. SLF4J Logger Documentation
6. SQLite Official Documentation
7. Java SQLite Tutorial | Baeldung