Lab -3

(Main.java)

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.stage.Stage;

public class Main extends Application {

@Override

public void start(Stage primaryStage) {

LoginPage loginPage = new LoginPage(primaryStage);

primaryStage.setTitle("HR Management - Parmjeet");

primaryStage.setScene(new Scene(loginPage.getView(), 400, 300));

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

(LoginPage.java)

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.\*;

import javafx.stage.Stage;

public class LoginPage {

private VBox view;

private TextField emailField;

private PasswordField passField;

public LoginPage(Stage stage) {

view = new VBox(10);

view.setPadding(new Insets(20));

emailField = new TextField();

emailField.setPromptText("Email");

passField = new PasswordField();

passField.setPromptText("Password");

Button loginBtn = new Button("Login");

Label messageLabel = new Label();

loginBtn.setOnAction(e -> {

EmployeeData.EmployeeCredential employee = LoginDAO.getEmployee(emailField.getText(), passField.getText());

if (employee != null) {

DashboardPage dashboard = new DashboardPage(stage, employee.getName() + " (" + employee.getRole() + ")");

stage.setScene(new Scene(dashboard.getView(), 500, 300));

} else {

messageLabel.setText("Invalid email or password");

}

});

view.getChildren().addAll(new Label("Login Page"), emailField, passField, loginBtn, messageLabel);

}

public VBox getView() {

return view;

}

}

(LoginDAO.java)

public class LoginDAO {

public static boolean validate(String email, String password) {

// Check against sample employee data

EmployeeData.EmployeeCredential employee = EmployeeData.validateLogin(email, password);

return employee != null;

}

public static EmployeeData.EmployeeCredential getEmployee(String email, String password) {

return EmployeeData.validateLogin(email, password);

}

}

(EmployeePage.java)

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.\*;

import javafx.stage.Stage;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

public class EmployeePage {

private VBox view;

private TableView<EmployeeData.EmployeeCredential> tableView;

private ObservableList<EmployeeData.EmployeeCredential> employeeList;

private TextField nameField, emailField, passwordField, salaryField;

private ComboBox<String> roleComboBox;

public EmployeePage(Stage stage) {

view = new VBox(10);

view.setPadding(new Insets(20));

Label title = new Label("Employee Panel - Employee Management");

title.setStyle("-fx-font-size: 18px; -fx-font-weight: bold;");

// Initialize employee list

employeeList = FXCollections.observableArrayList(EmployeeData.getSampleEmployees());

// Create form fields

createFormFields();

// Create table

createTableView();

// Create buttons

HBox buttonBox = createButtons(stage);

view.getChildren().addAll(title, createFormLayout(), buttonBox, tableView);

}

private void createFormFields() {

nameField = new TextField();

nameField.setPromptText("Name");

emailField = new TextField();

emailField.setPromptText("Email");

passwordField = new TextField();

passwordField.setPromptText("Password");

salaryField = new TextField();

salaryField.setPromptText("Salary");

roleComboBox = new ComboBox<>();

roleComboBox.getItems().addAll("Admin", "Employee");

roleComboBox.setPromptText("Select Role");

}

private GridPane createFormLayout() {

GridPane form = new GridPane();

form.setHgap(10);

form.setVgap(10);

form.add(new Label("Name:"), 0, 0);

form.add(nameField, 1, 0);

form.add(new Label("Email:"), 0, 1);

form.add(emailField, 1, 1);

form.add(new Label("Password:"), 0, 2);

form.add(passwordField, 1, 2);

form.add(new Label("Role:"), 0, 3);

form.add(roleComboBox, 1, 3);

form.add(new Label("Salary:"), 0, 4);

form.add(salaryField, 1, 4);

return form;

}

private void createTableView() {

tableView = new TableView<>();

TableColumn<EmployeeData.EmployeeCredential, String> nameCol = new TableColumn<>("Name");

nameCol.setCellValueFactory(new PropertyValueFactory<>("name"));

TableColumn<EmployeeData.EmployeeCredential, String> emailCol = new TableColumn<>("Email");

emailCol.setCellValueFactory(new PropertyValueFactory<>("email"));

TableColumn<EmployeeData.EmployeeCredential, String> roleCol = new TableColumn<>("Role");

roleCol.setCellValueFactory(new PropertyValueFactory<>("role"));

TableColumn<EmployeeData.EmployeeCredential, Double> salaryCol = new TableColumn<>("Salary");

salaryCol.setCellValueFactory(new PropertyValueFactory<>("salary"));

tableView.getColumns().addAll(nameCol, emailCol, roleCol, salaryCol);

tableView.setItems(employeeList);

// Handle row selection

tableView.getSelectionModel().selectedItemProperty().addListener((obs, oldSelection, newSelection) -> {

if (newSelection != null) {

populateForm(newSelection);

}

});

}

private void populateForm(EmployeeData.EmployeeCredential employee) {

nameField.setText(employee.getName());

emailField.setText(employee.getEmail());

passwordField.setText(employee.getPassword());

roleComboBox.setValue(employee.getRole());

salaryField.setText(String.valueOf(employee.getSalary()));

}

private void clearForm() {

nameField.clear();

emailField.clear();

passwordField.clear();

roleComboBox.setValue(null);

salaryField.clear();

}

private HBox createButtons(Stage stage) {

Button createBtn = new Button("Create");

Button updateBtn = new Button("Update");

Button deleteBtn = new Button("Delete");

Button viewBtn = new Button("View All");

Button logoutBtn = new Button("Logout");

Button backBtn = new Button("Back to Dashboard");

createBtn.setOnAction(e -> createEmployee());

updateBtn.setOnAction(e -> updateEmployee());

deleteBtn.setOnAction(e -> deleteEmployee());

viewBtn.setOnAction(e -> refreshTable());

logoutBtn.setOnAction(e -> {

LoginPage loginPage = new LoginPage(stage);

stage.setScene(new Scene(loginPage.getView(), 400, 300));

});

backBtn.setOnAction(e -> {

DashboardPage dashboard = new DashboardPage(stage, "Employee");

stage.setScene(new Scene(dashboard.getView(), 500, 300));

});

HBox buttonBox = new HBox(10);

buttonBox.getChildren().addAll(createBtn, updateBtn, deleteBtn, viewBtn, logoutBtn, backBtn);

return buttonBox;

}

private void createEmployee() {

if (validateForm()) {

EmployeeData.EmployeeCredential newEmployee = new EmployeeData.EmployeeCredential(

emailField.getText(),

passwordField.getText(),

nameField.getText(),

roleComboBox.getValue(),

Double.parseDouble(salaryField.getText())

);

employeeList.add(newEmployee);

clearForm();

showAlert("Success", "Employee created successfully!");

}

}

private void updateEmployee() {

EmployeeData.EmployeeCredential selected = tableView.getSelectionModel().getSelectedItem();

if (selected != null && validateForm()) {

int index = employeeList.indexOf(selected);

EmployeeData.EmployeeCredential updated = new EmployeeData.EmployeeCredential(

emailField.getText(),

passwordField.getText(),

nameField.getText(),

roleComboBox.getValue(),

Double.parseDouble(salaryField.getText())

);

employeeList.set(index, updated);

clearForm();

showAlert("Success", "Employee updated successfully!");

} else {

showAlert("Error", "Please select an employee to update!");

}

}

private void deleteEmployee() {

EmployeeData.EmployeeCredential selected = tableView.getSelectionModel().getSelectedItem();

if (selected != null) {

employeeList.remove(selected);

clearForm();

showAlert("Success", "Employee deleted successfully!");

} else {

showAlert("Error", "Please select an employee to delete!");

}

}

private void refreshTable() {

tableView.refresh();

}

private boolean validateForm() {

if (nameField.getText().isEmpty() || emailField.getText().isEmpty() ||

passwordField.getText().isEmpty() || roleComboBox.getValue() == null ||

salaryField.getText().isEmpty()) {

showAlert("Error", "Please fill in all fields!");

return false;

}

try {

Double.parseDouble(salaryField.getText());

} catch (NumberFormatException e) {

showAlert("Error", "Please enter a valid salary amount!");

return false;

}

return true;

}

private void showAlert(String title, String message) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

public VBox getView() {

return view;

}

}

(employeedata.java)

import java.util.ArrayList;

import java.util.List;

public class EmployeeData {

public static class EmployeeCredential {

private String email;

private String password;

private String name;

private String role;

private double salary;

public EmployeeCredential(String email, String password, String name, String role, double salary) {

this.email = email;

this.password = password;

this.name = name;

this.role = role;

this.salary = salary;

}

public String getEmail() { return email; }

public String getPassword() { return password; }

public String getName() { return name; }

public String getRole() { return role; }

public double getSalary() { return salary; }

}

public static List<EmployeeCredential> getSampleEmployees() {

List<EmployeeCredential> employees = new ArrayList<>();

employees.add(new EmployeeCredential("harman@company.com", "admin123", "Harman Admin", "Admin", 80000));

employees.add(new EmployeeCredential("navdeep@company.com", "nav123", "Navdeep Singh", "Employee", 55000));

employees.add(new EmployeeCredential("harjot@company.com", "harjot123", "harjot", "Employee", 60000));

employees.add(new EmployeeCredential("chmkila@company.com", "chamkila123", "chamkila", "Employee", 52000));

return employees;

}

public static EmployeeCredential validateLogin(String email, String password) {

for (EmployeeCredential emp : getSampleEmployees()) {

if (emp.getEmail().equals(email) && emp.getPassword().equals(password)) {

return emp;

}

}

return null;

}

}

(Employee.java)

public class Employee {

private int id;

private String name;

private double salary;

public Employee(int id, String name, double salary) {

this.id = id;

this.name = name;

this.salary = salary;

}

public double calculateYearlySalary() {

return salary \* 12;

}

}

(Dashboardpage.java)

public class Employee {

private int id;

private String name;

private double salary;

public Employee(int id, String name, double salary) {

this.id = id;

this.name = name;

this.salary = salary;

}

public double calculateYearlySalary() {

return salary \* 12;

}

}

(AdminPage.java)

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.\*;

import javafx.stage.Stage;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

public class AdminPage {

private VBox view;

private TableView<EmployeeData.EmployeeCredential> tableView;

private ObservableList<EmployeeData.EmployeeCredential> employeeList;

private TextField nameField, emailField, passwordField, salaryField;

private ComboBox<String> roleComboBox;

public AdminPage(Stage stage) {

view = new VBox(10);

view.setPadding(new Insets(20));

Label title = new Label("Admin Panel - Employee Management");

title.setStyle("-fx-font-size: 18px; -fx-font-weight: bold;");

// Initialize employee list

employeeList = FXCollections.observableArrayList(EmployeeData.getSampleEmployees());

// Create form fields

createFormFields();

// Create table

createTableView();

// Create buttons

HBox buttonBox = createButtons(stage);

view.getChildren().addAll(title, createFormLayout(), buttonBox, tableView);

}

private void createFormFields() {

nameField = new TextField();

nameField.setPromptText("Name");

emailField = new TextField();

emailField.setPromptText("Email");

passwordField = new TextField();

passwordField.setPromptText("Password");

salaryField = new TextField();

salaryField.setPromptText("Salary");

roleComboBox = new ComboBox<>();

roleComboBox.getItems().addAll("Admin", "Employee");

roleComboBox.setPromptText("Select Role");

}

private GridPane createFormLayout() {

GridPane form = new GridPane();

form.setHgap(10);

form.setVgap(10);

form.add(new Label("Name:"), 0, 0);

form.add(nameField, 1, 0);

form.add(new Label("Email:"), 0, 1);

form.add(emailField, 1, 1);

form.add(new Label("Password:"), 0, 2);

form.add(passwordField, 1, 2);

form.add(new Label("Role:"), 0, 3);

form.add(roleComboBox, 1, 3);

form.add(new Label("Salary:"), 0, 4);

form.add(salaryField, 1, 4);

return form;

}

private void createTableView() {

tableView = new TableView<>();

TableColumn<EmployeeData.EmployeeCredential, String> nameCol = new TableColumn<>("Name");

nameCol.setCellValueFactory(new PropertyValueFactory<>("name"));

TableColumn<EmployeeData.EmployeeCredential, String> emailCol = new TableColumn<>("Email");

emailCol.setCellValueFactory(new PropertyValueFactory<>("email"));

TableColumn<EmployeeData.EmployeeCredential, String> roleCol = new TableColumn<>("Role");

roleCol.setCellValueFactory(new PropertyValueFactory<>("role"));

TableColumn<EmployeeData.EmployeeCredential, Double> salaryCol = new TableColumn<>("Salary");

salaryCol.setCellValueFactory(new PropertyValueFactory<>("salary"));

tableView.getColumns().addAll(nameCol, emailCol, roleCol, salaryCol);

tableView.setItems(employeeList);

// Handle row selection

tableView.getSelectionModel().selectedItemProperty().addListener((obs, oldSelection, newSelection) -> {

if (newSelection != null) {

populateForm(newSelection);

}

});

}

private void populateForm(EmployeeData.EmployeeCredential employee) {

nameField.setText(employee.getName());

emailField.setText(employee.getEmail());

passwordField.setText(employee.getPassword());

roleComboBox.setValue(employee.getRole());

salaryField.setText(String.valueOf(employee.getSalary()));

}

private void clearForm() {

nameField.clear();

emailField.clear();

passwordField.clear();

roleComboBox.setValue(null);

salaryField.clear();

}

private HBox createButtons(Stage stage) {

Button createBtn = new Button("Create");

Button updateBtn = new Button("Update");

Button deleteBtn = new Button("Delete");

Button viewBtn = new Button("View All");

Button logoutBtn = new Button("Logout");

Button backBtn = new Button("Back to Dashboard");

createBtn.setOnAction(e -> createEmployee());

updateBtn.setOnAction(e -> updateEmployee());

deleteBtn.setOnAction(e -> deleteEmployee());

viewBtn.setOnAction(e -> refreshTable());

logoutBtn.setOnAction(e -> {

LoginPage loginPage = new LoginPage(stage);

stage.setScene(new Scene(loginPage.getView(), 400, 300));

});

backBtn.setOnAction(e -> {

DashboardPage dashboard = new DashboardPage(stage, "Admin");

stage.setScene(new Scene(dashboard.getView(), 500, 300));

});

HBox buttonBox = new HBox(10);

buttonBox.getChildren().addAll(createBtn, updateBtn, deleteBtn, viewBtn, logoutBtn, backBtn);

return buttonBox;

}

private void createEmployee() {

if (validateForm()) {

EmployeeData.EmployeeCredential newEmployee = new EmployeeData.EmployeeCredential(

emailField.getText(),

passwordField.getText(),

nameField.getText(),

roleComboBox.getValue(),

Double.parseDouble(salaryField.getText())

);

employeeList.add(newEmployee);

clearForm();

showAlert("Success", "Employee created successfully!");

}

}

private void updateEmployee() {

EmployeeData.EmployeeCredential selected = tableView.getSelectionModel().getSelectedItem();

if (selected != null && validateForm()) {

int index = employeeList.indexOf(selected);

EmployeeData.EmployeeCredential updated = new EmployeeData.EmployeeCredential(

emailField.getText(),

passwordField.getText(),

nameField.getText(),

roleComboBox.getValue(),

Double.parseDouble(salaryField.getText())

);

employeeList.set(index, updated);

clearForm();

showAlert("Success", "Employee updated successfully!");

} else {

showAlert("Error", "Please select an employee to update!");

}

}

private void deleteEmployee() {

EmployeeData.EmployeeCredential selected = tableView.getSelectionModel().getSelectedItem();

if (selected != null) {

employeeList.remove(selected);

clearForm();

showAlert("Success", "Employee deleted successfully!");

} else {

showAlert("Error", "Please select an employee to delete!");

}

}

private void refreshTable() {

tableView.refresh();

}

private boolean validateForm() {

if (nameField.getText().isEmpty() || emailField.getText().isEmpty() ||

passwordField.getText().isEmpty() || roleComboBox.getValue() == null ||

salaryField.getText().isEmpty()) {

showAlert("Error", "Please fill in all fields!");

return false;

}

try {

Double.parseDouble(salaryField.getText());

} catch (NumberFormatException e) {

showAlert("Error", "Please enter a valid salary amount!");

return false;

}

return true;

}

private void showAlert(String title, String message) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

public VBox getView() {

return view;

}

}

Employees

A screenshot of a computer

AI-generated content may be incorrect.

Employ updated

A screenshot of a computer

AI-generated content may be incorrect.

Salary updated

A screenshot of a computer

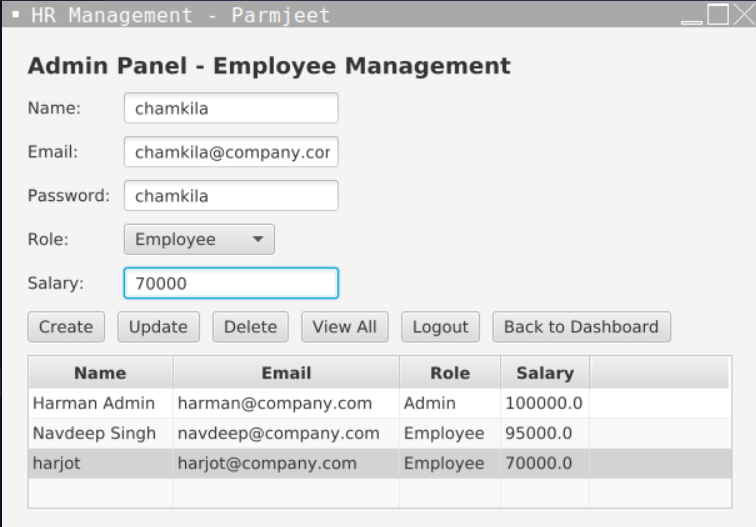
AI-generated content may be incorrect.

Employee Deleted

A screenshot of a computer

AI-generated content may be incorrect.

Creating new employee



Chamkila added as a new employee after delete

A screenshot of a computer

AI-generated content may be incorrect.