**Comsats University Islamabad, Lahore Campus**

****

**OOP Project Code**

**Submitted to:**

Sir Shahid Bhatti

**Submitted By:**SP24-BSE-107 Samiullah Ashfaq

SP24-BSE-125 Zain Yaqoob

SP24-BSE-068 Muhammad Asad

**Code for Hostel Management System**

**Main.java**

package com.example.project;

import com.example.project.MenuStage;

import javafx.application.Application;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.Pane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.stage.Stage;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

public class Main extends Application {

@Override

public void start(Stage primaryStage) {

// Root Pane

Pane root = new Pane();

root.setStyle("-fx-background-color: linear-gradient(to right, #0a0f2e, #a75ebc);");

// Title Label

Label title = new Label("HOSTEL MANAGEMENT SYSTEM");

title.setFont(Font.font("Arial", 36));

title.setTextFill(Color.WHITE);

title.setLayoutX(150);

title.setLayoutY(50);

// Login Form Container

VBox loginBox = new VBox(15);

loginBox.setAlignment(Pos.CENTER);

loginBox.setPrefWidth(300);

loginBox.setLayoutX(250);

loginBox.setLayoutY(150);

// Username Field

Label usernameLabel = new Label("Username");

usernameLabel.setFont(Font.font("Arial", 16));

usernameLabel.setTextFill(Color.WHITE);

TextField userField = new TextField();

userField.setPromptText("Enter Username");

// Password Field

Label passwordLabel = new Label("Password");

passwordLabel.setFont(Font.font("Arial", 16));

passwordLabel.setTextFill(Color.WHITE);

PasswordField passwordField = new PasswordField();

passwordField.setPromptText("Enter Password");

// Login Button

Button loginButton = new Button("Log In");

loginButton.setStyle("-fx-background-color: #3b7ef8; -fx-text-fill: white; -fx-font-size: 14px;");

// Sign Up Button

Button signUpButton = new Button("Sign Up");

signUpButton.setStyle("-fx-background-color: #45c96b; -fx-text-fill: white; -fx-font-size: 14px;");

// Add elements to loginBox

loginBox.getChildren().addAll(usernameLabel, userField, passwordLabel, passwordField, loginButton, signUpButton);

root.getChildren().addAll(title, loginBox);

loginButton.setOnAction(e -> {

String username = userField.getText();

String password = passwordField.getText();

// Check if username or password is empty

if (username.isEmpty() || password.isEmpty()) {

showAlert("Input Error", "Username and password fields cannot be empty", Alert.AlertType.WARNING);

return;

}

try {

java.io.File file = new java.io.File("user.txt");

// Check if file exists

if (!file.exists()) {

showAlert("File Error", "User data file does not exist", Alert.AlertType.ERROR);

return;

}

// Check if file is empty

if (file.length() == 0) {

showAlert("File Error", "User data file is empty", Alert.AlertType.ERROR);

return;

}

// Read and validate credentials

try (BufferedReader reader = new BufferedReader(new FileReader(file))) {

String line;

boolean userFound = false;

while ((line = reader.readLine()) != null) {

String[] credentials = line.split(",");

if (credentials.length >= 2 && credentials[0].equalsIgnoreCase(username) && credentials[1].equals(password)) {

userFound = true;

// Open MenuStage and close the login stage

Manage manageStage = new Manage();

manageStage.show(primaryStage);

return;

}

}

if (!userFound) {

showAlert("Login Failed", "Invalid username or password", Alert.AlertType.ERROR);

}

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "An unexpected error occurred while accessing the user data file", Alert.AlertType.ERROR);

}

});

signUpButton.setOnAction(e -> {

String username = userField.getText();

String password = passwordField.getText();

// Check if username or password is empty

if (username.isEmpty() || password.isEmpty()) {

showAlert("Input Error", "Username and password fields cannot be empty", Alert.AlertType.WARNING);

return;

}

try (BufferedWriter writer = new BufferedWriter(new FileWriter("user.txt", true))) {

String data = username + "," + password + System.lineSeparator();

writer.write(data);

showAlert("Sign Up Successful", "You can now log in with your credentials.", Alert.AlertType.INFORMATION);

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Unable to save user data", Alert.AlertType.ERROR);

}

});

// Scene

Scene scene = new Scene(root, 800, 600);

primaryStage.setScene(scene);

primaryStage.setTitle("Hostel Management System");

primaryStage.show();

}

private void showAlert(String title, String content, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(content);

alert.showAndWait();

}

public static void main(String[] args) {

launch(args);

}

}

**Manage.java**

package com.example.project;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.layout.Pane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.Text;

import javafx.stage.Stage;

public class Manage {

public void show(Stage stage) {

// Root Pane

Pane root = new Pane();

root.setStyle("-fx-background-color: linear-gradient(to right, #0a0f2e, #a75ebc);");

// Title Label

Text title = new Text("HOSTEL MANAGEMENT SYSTEM");

title.setFont(Font.font("Arial", 30));

title.setFill(Color.WHITE);

title.setX(150);

title.setY(50);

// VBox for Buttons

VBox buttonBox = new VBox(20);

buttonBox.setAlignment(Pos.CENTER);

buttonBox.setPrefWidth(300);

buttonBox.setLayoutX(250);

buttonBox.setLayoutY(150);

// Manage Students Button

Button manageStudentsButton = new Button("Manage Students");

manageStudentsButton.setStyle("-fx-background-color: #3b7ef8; -fx-text-fill: white; -fx-font-size: 16px; -fx-pref-width: 200px; -fx-pref-height: 50px;");

// Manage Employees Button

Button manageEmployeesButton = new Button("Manage Employees");

manageEmployeesButton.setStyle("-fx-background-color: #3b7ef8; -fx-text-fill: white; -fx-font-size: 16px; -fx-pref-width: 200px; -fx-pref-height: 50px;");

// Logout Button

Button logoutButton = new Button("Log Out");

logoutButton.setStyle("-fx-background-color: #e84545; -fx-text-fill: white; -fx-font-size: 14px; -fx-pref-width: 100px; -fx-pref-height: 30px;");

logoutButton.setLayoutX(650);

logoutButton.setLayoutY(20);

manageStudentsButton.setOnAction(e -> {

stage.close();

new MenuStage(stage).show(); // Pass the current stage to MenuStage

});

manageEmployeesButton.setOnAction(e -> {

stage.close();

new EmployeeStage(stage).show(); // Pass the current stage to MenuStage

});

// Logout Button Action

logoutButton.setOnAction(e -> {

new Main().start(stage); // Redirect to Main Login Screen

showAlert("Logged Out", "You have logged out successfully.", Alert.AlertType.INFORMATION);

});

// Add Buttons to VBox

buttonBox.getChildren().addAll(manageStudentsButton, manageEmployeesButton);

root.getChildren().addAll(title, buttonBox, logoutButton);

// Scene

Scene scene = new Scene(root, 800, 600);

stage.setScene(scene);

stage.setTitle("Manage");

stage.show();

}

private void showAlert(String title, String content, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(content);

alert.showAndWait();

}

}

**MenuStage.java**

package com.example.project;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.scene.text.Font;

import javafx.stage.Stage;

public class MenuStage {

private final Stage previousStage; // Store reference to the previous stage

// Constructor to accept the previous stage (Manage stage)

public MenuStage(Stage previousStage) {

this.previousStage = previousStage;

}

public void show() {

Stage menuStage = new Stage();

// Root layout

VBox root = new VBox(20);

root.setStyle("-fx-background-color: linear-gradient(to bottom right, #0f0c29, #302b63, #24243e);");

root.setPadding(new Insets(20));

// Top Bar Layout for Back Button

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_RIGHT);

topBar.setPadding(new Insets(10));

// Back Button

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #e84545; -fx-text-fill: white; -fx-font-size: 14px; -fx-padding: 10px 20px; -fx-background-radius: 8px;");

backButton.setOnAction(e -> {

menuStage.close(); // Close the current stage

new Manage().show(previousStage); // Show the Manage stage

});

// Add Back Button to the Top Bar

topBar.getChildren().add(backButton);

// Button style

String buttonStyle = "-fx-background-color: linear-gradient(to bottom right, #000428, #004e92);"

+ "-fx-text-fill: white;"

+ "-fx-font-size: 16px;"

+ "-fx-font-weight: bold;"

+ "-fx-padding: 15px 30px;"

+ "-fx-background-radius: 10;"

+ "-fx-pref-width: 300;";

// Buttons

Button addButton = new Button("ADD A NEW STUDENT RECORD");

addButton.setStyle(buttonStyle);

addButton.setFont(Font.font("Arial", 14));

addButton.setOnAction(e -> handleAddStudentAction(menuStage));

Button updateButton = new Button("UPDATE A STUDENT RECORD");

updateButton.setStyle(buttonStyle);

updateButton.setFont(Font.font("Arial", 14));

updateButton.setOnAction(e -> handleUpdateStudentAction(menuStage));

Button deleteButton = new Button("DELETE A STUDENT RECORD");

deleteButton.setStyle(buttonStyle);

deleteButton.setFont(Font.font("Arial", 14));

deleteButton.setOnAction(e -> handleDeleteStudentAction(menuStage));

Button viewLiveInButton = new Button("VIEW THE LIVE-IN STUDENT DETAILS");

viewLiveInButton.setStyle(buttonStyle);

viewLiveInButton.setFont(Font.font("Arial", 14));

viewLiveInButton.setOnAction(e -> handleViewLiveInStudents(menuStage));

// Add components to the root layout

root.getChildren().addAll(topBar, addButton, updateButton, deleteButton, viewLiveInButton);

// Scene and stage setup

Scene scene = new Scene(root, 800, 600);

menuStage.setScene(scene);

menuStage.setTitle("Menu");

menuStage.show();

}

private void handleAddStudentAction(Stage menuStage) {

menuStage.close();

new AddStudentStage(menuStage).show();

}

private void handleUpdateStudentAction(Stage menuStage) {

menuStage.close();

new UpdateStudentStage(menuStage).show();

}

private void handleDeleteStudentAction(Stage menuStage) {

menuStage.close();

new DeleteStudentStage(menuStage).show();

}

private void handleViewLiveInStudents(Stage menuStage) {

menuStage.close();

new ViewLiveInStudentsStage(menuStage).show();

}

}

**Student.java**

package com.example.project;

public class Student {

private String id;

private String name;

private String guardianName;

private String age;

private String cnic;

public Student(String id, String name, String guardianName, String age, String cnic) {

this.id = id;

this.name = name;

this.guardianName = guardianName;

this.age = age;

this.cnic = cnic;

}

public String getId() {

return id;

}

public String getName() {

return name;

}

public String getGuardianName() {

return guardianName;

}

public String getAge() {

return age;

}

public String getCnic() {

return cnic;

}

}

**AddStudent.java**

package com.example.project;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.VBox;

import javafx.scene.layout.HBox;

import javafx.stage.Stage;

import java.io.\*;

public class AddStudentStage {

private Stage menuStage; // Reference to the MenuStage

public AddStudentStage(Stage menuStage) {

this.menuStage = menuStage; // Save reference to menuStage passed from MenuStage

}

public void show() {

// Create a new stage for adding student details

Stage addStudentStage = new Stage();

// Create a VBox layout to organize the form fields

VBox layout = new VBox(15); // 15px spacing between fields

layout.setStyle("-fx-background-color: linear-gradient(to bottom right, #0f0c29, #302b63, #24243e); -fx-padding: 20px;");

layout.setAlignment(Pos.CENTER);

// Define universal styles for the UI elements

String labelStyle = "-fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;";

String textFieldStyle = "-fx-background-color: white; -fx-border-radius: 8px; -fx-padding: 5px; -fx-font-size: 12px;";

String buttonStyle = "-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;";

// Create Labels and TextFields for Student ID, Name, Guardian Name, Age, and CNIC

Label studentIdLabel = new Label("Student ID:");

studentIdLabel.setStyle(labelStyle);

TextField studentIdField = new TextField();

studentIdField.setStyle(textFieldStyle);

Label nameLabel = new Label("Name:");

nameLabel.setStyle(labelStyle);

TextField nameField = new TextField();

nameField.setStyle(textFieldStyle);

Label guardianNameLabel = new Label("Guardian Name:");

guardianNameLabel.setStyle(labelStyle);

TextField guardianNameField = new TextField();

guardianNameField.setStyle(textFieldStyle);

Label ageLabel = new Label("Age:");

ageLabel.setStyle(labelStyle);

TextField ageField = new TextField();

ageField.setStyle(textFieldStyle);

Label cnicLabel = new Label("CNIC:");

cnicLabel.setStyle(labelStyle);

TextField cnicField = new TextField();

cnicField.setStyle(textFieldStyle);

// Create an "Add" button to save the student details

Button addButton = new Button("Add");

addButton.setStyle(buttonStyle);

// Add hover effect for the "Add" button

addButton.setOnMouseEntered(e -> addButton.setStyle("-fx-background-color: #45a049; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;"));

addButton.setOnMouseExited(e -> addButton.setStyle(buttonStyle));

// Add event handler for the "Add" button

addButton.setOnAction(e -> {

String studentId = studentIdField.getText();

String name = nameField.getText();

String guardianName = guardianNameField.getText();

String age = ageField.getText();

String cnic = cnicField.getText();

if (studentId.isEmpty() || name.isEmpty() || guardianName.isEmpty() || age.isEmpty() || cnic.isEmpty()) {

showAlert("Error", "All fields are required", Alert.AlertType.ERROR);

return;

}

try {

File file = new File("Students.txt");

// Check if Student ID already exists

if (file.exists()) {

try (BufferedReader reader = new BufferedReader(new FileReader(file))) {

String line;

while ((line = reader.readLine()) != null) {

String[] details = line.split(",");

if (details.length >= 5 && details[0].equals(studentId)) {

showAlert("Error", "Student ID already exists", Alert.AlertType.ERROR);

return;

}

}

}

}

// Write the student's information to the Students.txt file

try (BufferedWriter writer = new BufferedWriter(new FileWriter(file, true))) {

// Create a string with student details

String studentData = studentId + "," + name + "," + guardianName + "," + age + "," + cnic + System.lineSeparator();

// Write to the file

writer.write(studentData);

showAlert("Success", "The student has been added successfully.", Alert.AlertType.INFORMATION);

// Clear the fields for new student entry, keeping the AddStudentStage open

studentIdField.clear();

nameField.clear();

guardianNameField.clear();

ageField.clear();

cnicField.clear();

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to add student. Please try again.", Alert.AlertType.ERROR);

}

});

// Create a "Display" button to show the live-in students

Button displayButton = new Button("Display");

displayButton.setStyle(buttonStyle);

// Add hover effect for the "Display" button

displayButton.setOnMouseEntered(e -> displayButton.setStyle("-fx-background-color: #45a049; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;"));

displayButton.setOnMouseExited(e -> displayButton.setStyle(buttonStyle));

// Add event handler for the "Display" button

displayButton.setOnAction(e -> {

addStudentStage.close();

ViewLiveInStudentsStage viewStage = new ViewLiveInStudentsStage(addStudentStage); // Passing the current stage as the previous stage

viewStage.show();

});

// Create an HBox to position the Add and Display buttons

HBox buttonBox = new HBox(10); // 10px spacing between buttons

buttonBox.setAlignment(Pos.CENTER);

buttonBox.getChildren().addAll(addButton, displayButton); // Add both buttons to the HBox

// Back Button (Top Right Corner)

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #4569a0; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

backButton.setOnAction(e -> {

addStudentStage.close(); // Close the AddStudentStage

menuStage.show(); // Show the MenuStage again

});

// Create an HBox to position the Back button at the top-right

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_RIGHT); // Align the button to the top-right

topBar.getChildren().add(backButton); // Add the Back button to the top bar

// Add all components to the layout

layout.getChildren().addAll(topBar, studentIdLabel, studentIdField, nameLabel, nameField, guardianNameLabel, guardianNameField, ageLabel, ageField, cnicLabel, cnicField, buttonBox);

// Create and show the scene for the add student stage

Scene scene = new Scene(layout, 600, 600); // Adjusted scene size

addStudentStage.setScene(scene);

addStudentStage.setTitle("Add New Student");

addStudentStage.show();

}

private void showAlert(String title, String content, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(content);

alert.showAndWait();

}

}

UpdateStudentStage.java

package com.example.project;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.VBox;

import javafx.scene.layout.HBox;

import javafx.stage.Stage;

import java.io.\*;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.List;

public class UpdateStudentStage {

private Stage menuStage; // Reference to the MenuStage

public UpdateStudentStage(Stage menuStage) {

this.menuStage = menuStage; // Save reference to menuStage passed from MenuStage

}

public void show() {

// Create a new stage for updating student details

Stage updateStudentStage = new Stage();

// Create a VBox layout to organize the form fields

VBox layout = new VBox(15); // 15px spacing between fields

layout.setStyle("-fx-background-color: linear-gradient(to bottom right, #0f0c29, #302b63, #24243e); -fx-padding: 20px;");

layout.setAlignment(Pos.CENTER);

// Define a universal label style

String labelStyle = "-fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;";

// Label and TextField to search by student's ID

Label searchLabel = new Label("Enter Student ID to Update:");

searchLabel.setStyle(labelStyle);

TextField searchField = new TextField();

searchField.setPromptText("Student ID");

// Button to search the student

Button searchButton = new Button("Search");

searchButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

// Create fields for displaying and updating student details (now editable)

Label nameLabel = new Label("Name:");

nameLabel.setStyle(labelStyle);

TextField nameField = new TextField();

Label guardianNameLabel = new Label("Guardian Name:");

guardianNameLabel.setStyle(labelStyle);

TextField guardianNameField = new TextField();

Label ageLabel = new Label("Age:");

ageLabel.setStyle(labelStyle);

TextField ageField = new TextField();

Label cnicLabel = new Label("CNIC:");

cnicLabel.setStyle(labelStyle);

TextField cnicField = new TextField();

// Create an "Update" button to save the updated student details

Button updateButton = new Button("Update");

updateButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

updateButton.setDisable(true); // Initially disabled until the student is found

// Create a "Display" button to view live information in a new stage

Button displayButton = new Button("Display");

displayButton.setStyle("-fx-background-color: #2196F3; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

// Event handler for search button

searchButton.setOnAction(e -> handleSearchAction(searchField, nameField, guardianNameField, ageField, cnicField, updateButton));

// Event handler for update button

updateButton.setOnAction(e -> handleUpdateAction(searchField, nameField, guardianNameField, ageField, cnicField, updateButton));

// Event handler for display button

displayButton.setOnAction(e -> {

updateStudentStage.close();

ViewLiveInStudentsStage viewLiveInStudentStage = new ViewLiveInStudentsStage(updateStudentStage);

viewLiveInStudentStage.show();

});

// Create an HBox for the Update and Display buttons

HBox buttonBox = new HBox(10); // 10px spacing between buttons

buttonBox.setAlignment(Pos.CENTER);

buttonBox.getChildren().addAll(updateButton, displayButton);

// Back Button (Top Right Corner)

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #4569a0; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

backButton.setOnAction(e -> {

updateStudentStage.close(); // Close the UpdateStudentStage

menuStage.show(); // Show the MenuStage again

});

// Create an HBox to position the Back button at the top-right

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_RIGHT); // Align the button to the top-right

topBar.getChildren().add(backButton); // Add the Back button to the top bar

// Add all components to the layout

layout.getChildren().addAll(topBar, searchLabel, searchField, searchButton,

nameLabel, nameField,

guardianNameLabel, guardianNameField,

ageLabel, ageField,

cnicLabel, cnicField,

buttonBox);

// Create and show the scene for the update student stage

Scene scene = new Scene(layout, 600, 600);

updateStudentStage.setScene(scene);

updateStudentStage.setTitle("Update Student Record");

updateStudentStage.show();

}

private void handleSearchAction(TextField searchField, TextField nameField, TextField guardianNameField, TextField ageField, TextField cnicField, Button updateButton) {

String searchID = searchField.getText().trim();

if (searchID.isEmpty()) {

showAlert("Error", "Student ID is required", "Please enter a student ID to search.", Alert.AlertType.ERROR);

return;

}

try (BufferedReader reader = new BufferedReader(new FileReader("Students.txt"))) {

String line;

boolean found = false;

while ((line = reader.readLine()) != null) {

String[] studentData = line.split(",");

if (studentData[0].equals(searchID)) {

// Populate the fields with existing data

nameField.setText(studentData[1]);

guardianNameField.setText(studentData[2]);

ageField.setText(studentData[3]);

cnicField.setText(studentData[4]);

updateButton.setDisable(false); // Enable the update button

found = true;

break;

}

}

if (!found) {

showAlert("Student Not Found", "Student Not Found", "No student found with the ID: " + searchID, Alert.AlertType.ERROR);

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to Search", "There was an error searching for the student.", Alert.AlertType.ERROR);

}

}

private void handleUpdateAction(TextField searchField, TextField nameField, TextField guardianNameField, TextField ageField, TextField cnicField, Button updateButton) {

String searchID = searchField.getText().trim();

String newName = nameField.getText().trim();

String guardianName = guardianNameField.getText().trim();

String age = ageField.getText().trim();

String cnic = cnicField.getText().trim();

if (newName.isEmpty() || guardianName.isEmpty() || age.isEmpty() || cnic.isEmpty()) {

showAlert("Error", "All fields are required", "Please fill in all the fields.", Alert.AlertType.ERROR);

return;

}

try {

List<String> lines = Files.readAllLines(Paths.get("Students.txt"));

boolean updated = false;

for (int i = 0; i < lines.size(); i++) {

String[] studentData = lines.get(i).split(",");

if (studentData[0].equals(searchID)) {

lines.set(i, searchID + "," + newName + "," + guardianName + "," + age + "," + cnic);

updated = true;

break;

}

}

if (updated) {

Files.write(Paths.get("Students.txt"), lines);

showAlert("Success", "Student Updated", "The student's record has been updated successfully.", Alert.AlertType.INFORMATION);

// Clear the fields and keep the stage open for further updates

nameField.clear();

guardianNameField.clear();

ageField.clear();

cnicField.clear();

// Optionally, disable the update button again after update

updateButton.setDisable(true);

} else {

showAlert("Error", "Update Failed", "No matching student found to update.", Alert.AlertType.ERROR);

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to Update", "There was an error updating the student record.", Alert.AlertType.ERROR);

}

}

private void showAlert(String title, String header, String content, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(header);

alert.setContentText(content);

alert.showAndWait();

}

}

**DeleteStudentStage.java**

package com.example.project;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

public class DeleteStudentStage {

private Stage menuStage; // Reference to MenuStage

public DeleteStudentStage(Stage menuStage) {

this.menuStage = menuStage; // Save reference to menuStage passed from MenuStage

}

public void show() {

// Create a new stage for deleting a student

Stage deleteStudentStage = new Stage();

deleteStudentStage.setTitle("Delete Student");

// Create a VBox layout for the delete student form

VBox layout = new VBox(15); // 15px spacing between fields

layout.setStyle("-fx-background-color: linear-gradient(to bottom right, #0f0c29, #302b63, #24243e); -fx-padding: 20px;");

layout.setAlignment(Pos.CENTER);

// Define style for the UI elements

String labelStyle = "-fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;";

String textFieldStyle = "-fx-background-color: white; -fx-border-radius: 10px; -fx-padding: 10px; -fx-font-size: 14px;";

String buttonStyle = "-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;";

// Create a Label and TextField for student ID input

Label studentIdLabel = new Label("Enter Student ID to Delete:");

studentIdLabel.setStyle(labelStyle);

TextField studentIdField = new TextField();

studentIdField.setStyle(textFieldStyle);

// Create Labels to display student details (initially hidden)

Label nameLabel = new Label("Name: ");

nameLabel.setStyle(labelStyle);

Label guardianNameLabel = new Label("Guardian Name: ");

guardianNameLabel.setStyle(labelStyle);

Label ageLabel = new Label("Age: ");

ageLabel.setStyle(labelStyle);

Label cnicLabel = new Label("CNIC: ");

cnicLabel.setStyle(labelStyle);

// Initially disable the delete button (the display button will always be enabled)

Button deleteButton = new Button("Delete");

deleteButton.setStyle(buttonStyle);

deleteButton.setDisable(true); // Disable delete initially

Button displayButton = new Button("Display");

displayButton.setStyle(buttonStyle);

// Display button is always enabled

displayButton.setOnAction(e -> {deleteStudentStage.close();

ViewLiveInStudentsStage viewStage = new ViewLiveInStudentsStage(deleteStudentStage); // Passing the current stage as the previous stage

viewStage.show();}

);

// Back Button (Top Right Corner)

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #4569a0; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

backButton.setOnAction(e -> {

deleteStudentStage.close(); // Close the DeleteStudentStage

menuStage.show(); // Show the MenuStage again

});

// Create an HBox to position the Back button at the top-right

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_RIGHT); // Align the button to the top-right

topBar.getChildren().add(backButton); // Add the Back button to the top bar

// Add event handler for the "Search" button

Button searchButton = new Button("Search");

searchButton.setStyle(buttonStyle);

searchButton.setOnAction(e -> handleSearchAction(studentIdField, nameLabel, guardianNameLabel, ageLabel, cnicLabel, deleteButton, displayButton));

// Add event handler for the "Delete" button

deleteButton.setOnAction(e -> handleDeleteAction(studentIdField, deleteButton, displayButton, nameLabel, guardianNameLabel, ageLabel, cnicLabel));

// Create an HBox for the delete and display buttons

HBox buttonBox = new HBox(10); // 10px spacing between buttons

buttonBox.setAlignment(Pos.CENTER);

buttonBox.getChildren().addAll(displayButton, deleteButton);

// Add the input field, search button, labels, and buttons to the layout

layout.getChildren().addAll(topBar, studentIdLabel, studentIdField, searchButton, nameLabel, guardianNameLabel, ageLabel, cnicLabel, buttonBox);

// Create and show the scene for the delete student stage

Scene scene = new Scene(layout, 500, 500);

deleteStudentStage.setScene(scene);

deleteStudentStage.show();

}

private void handleSearchAction(TextField studentIdField, Label nameLabel, Label guardianNameLabel, Label ageLabel, Label cnicLabel, Button deleteButton, Button displayButton) {

String studentId = studentIdField.getText().trim();

if (studentId.isEmpty()) {

showAlert("Error", "Student ID is required", "Please enter the ID of the student to search.", Alert.AlertType.ERROR);

return;

}

// Search for student in the file

File studentFile = new File("Students.txt");

boolean studentFound = false;

try (BufferedReader reader = new BufferedReader(new FileReader(studentFile))) {

String line;

while ((line = reader.readLine()) != null) {

String[] studentData = line.split(",");

if (studentData.length >= 5 && studentData[0].equals(studentId)) {

// Display student details

nameLabel.setText("Name: " + studentData[1]);

guardianNameLabel.setText("Guardian Name: " + studentData[2]);

ageLabel.setText("Age: " + studentData[3]);

cnicLabel.setText("CNIC: " + studentData[4]);

// Enable delete button

deleteButton.setDisable(false);

studentFound = true;

break;

}

}

if (!studentFound) {

showAlert("Student Not Found", null, "No student found with the ID: " + studentId, Alert.AlertType.ERROR);

deleteButton.setDisable(true);

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to Search", "There was an error reading the student data.", Alert.AlertType.ERROR);

}

}

// Handle the deletion process when "Delete" is pressed

private void handleDeleteAction(TextField studentIdField, Button deleteButton, Button displayButton, Label nameLabel, Label guardianNameLabel, Label ageLabel, Label cnicLabel) {

String studentId = studentIdField.getText().trim();

if (studentId.isEmpty()) {

showAlert("Error", "Student ID is required", "Please enter the ID of the student to delete.", Alert.AlertType.ERROR);

return;

}

// Search for student in the file and delete the record

File studentFile = new File("Students.txt");

List<String> updatedContent = new ArrayList<>();

boolean studentFound = false;

try (BufferedReader reader = new BufferedReader(new FileReader(studentFile))) {

String line;

while ((line = reader.readLine()) != null) {

String[] studentData = line.split(",");

if (studentData.length >= 5 && studentData[0].equals(studentId)) {

studentFound = true; // Student found, skipping the record to delete it

continue;

}

updatedContent.add(line); // Add other students to the updated list

}

if (!studentFound) {

showAlert("Student Not Found", null, "No student found with the ID: " + studentId, Alert.AlertType.ERROR);

} else {

// Overwrite the file with the updated content (without the deleted student)

try (BufferedWriter writer = new BufferedWriter(new FileWriter(studentFile))) {

for (String lineContent : updatedContent) {

writer.write(lineContent);

writer.newLine();

}

}

showAlert("Success", "Student Deleted", "The student record has been successfully deleted.", Alert.AlertType.INFORMATION);

// Reset the labels and disable the buttons

nameLabel.setText("Name: ");

guardianNameLabel.setText("Guardian Name: ");

ageLabel.setText("Age: ");

cnicLabel.setText("CNIC: ");

studentIdField.clear();

deleteButton.setDisable(true);

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to Delete Student", "There was an error reading or writing the student data.", Alert.AlertType.ERROR);

}

}

private void showAlert(String title, String header, String content, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(header);

alert.setContentText(content);

alert.showAndWait();

}

}

**ViewLiveInStudentStage.java**

package com.example.project;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.VBox;

import javafx.scene.layout.HBox;

import javafx.stage.Stage;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class ViewLiveInStudentsStage {

private Stage previousStage; // Reference to the previous stage

public ViewLiveInStudentsStage(Stage previousStage) {

this.previousStage = previousStage; // Save reference to the previous stage passed from MenuStage or AddStudentStage

}

public void show() {

// Create a new stage to display the live-in students

Stage viewStage = new Stage();

viewStage.setTitle("Live-In Students");

// Create a VBox layout to hold the TableView

VBox layout = new VBox(10);

layout.setStyle("-fx-background-color: linear-gradient(to bottom right, #0f0c29, #302b63, #24243e); -fx-padding: 20px;");

layout.setAlignment(Pos.CENTER);

// Create the TableView and its columns

TableView<Student> tableView = new TableView<>();

// Create the columns for the TableView

TableColumn<Student, String> idColumn = new TableColumn<>("Student ID");

idColumn.setCellValueFactory(new PropertyValueFactory<>("id"));

TableColumn<Student, String> nameColumn = new TableColumn<>("Name");

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

TableColumn<Student, String> guardianNameColumn = new TableColumn<>("Guardian Name");

guardianNameColumn.setCellValueFactory(new PropertyValueFactory<>("guardianName"));

TableColumn<Student, String> ageColumn = new TableColumn<>("Age");

ageColumn.setCellValueFactory(new PropertyValueFactory<>("age"));

TableColumn<Student, String> cnicColumn = new TableColumn<>("CNIC");

cnicColumn.setCellValueFactory(new PropertyValueFactory<>("cnic"));

// Add columns to the TableView

tableView.getColumns().addAll(idColumn, nameColumn, guardianNameColumn, ageColumn, cnicColumn);

// Read the data from the Students.txt file and populate the TableView

try (BufferedReader reader = new BufferedReader(new FileReader("Students.txt"))) {

String line;

while ((line = reader.readLine()) != null) {

String[] studentData = line.split(",");

if (studentData.length == 5) { // Ensure all fields including ID are present

// Create a new Student object and add it to the TableView

Student student = new Student(studentData[0], studentData[1], studentData[2], studentData[3], studentData[4]);

tableView.getItems().add(student);

}

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to Load Students", "There was an error reading the student data.", Alert.AlertType.ERROR);

}

// Create a Back button (Top Right Corner)

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #4569a0; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

backButton.setOnAction(e -> {

viewStage.close(); // Close the ViewLiveInStudentsStage

previousStage.show(); // Show the previous stage (either MenuStage or AddStudentStage)

});

// Create an HBox to position the Back button at the top-right

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_RIGHT); // Align the button to the top-right

topBar.getChildren().add(backButton); // Add the Back button to the top bar

// Add all components to the layout

layout.getChildren().addAll(topBar, tableView);

// Create and show the scene for the view students stage

Scene scene = new Scene(layout, 800, 600);

viewStage.setScene(scene);

viewStage.show();

}

private void showAlert(String title, String header, String content, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(header);

alert.setContentText(content);

alert.showAndWait();

}

}

**Employee.java**

package com.example.project;

public class Employee {

private String employeeId;

private String name;

private String cnic;

private String age;

// Constructor to initialize the Employee object

public Employee(String employeeId, String name, String age, String cnic) {

this.employeeId = employeeId;

this.name = name;

this.age = age;

this.cnic = cnic;

}

// Getters for the fields

public String getEmployeeId() {

return employeeId;

}

public String getName() {

return name;

}

public String getCnic() {

return cnic;

}

public String getAge() {

return age;

}

}

**EmployeeStage.java**

package com.example.project;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.scene.text.Font;

import javafx.stage.Stage;

public class EmployeeStage {

private final Stage previousStage; // Store reference to the previous stage

// Constructor to accept the previous stage (Manage stage)

public EmployeeStage(Stage previousStage) {

this.previousStage = previousStage;

}

public void show() {

Stage employeeStage = new Stage();

// Root layout

VBox root = new VBox(20);

root.setStyle("-fx-background-color: linear-gradient(to bottom right, #0f0c29, #302b63, #24243e);");

root.setPadding(new Insets(20));

// Top Bar Layout for Back Button

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_RIGHT);

topBar.setPadding(new Insets(10));

// Back Button

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #e84545; -fx-text-fill: white; -fx-font-size: 14px; -fx-padding: 10px 20px; -fx-background-radius: 8px;");

backButton.setOnAction(e -> {

employeeStage.close(); // Close the current stage

new Manage().show(previousStage); // Show the Manage stage

});

// Add Back Button to the Top Bar

topBar.getChildren().add(backButton);

// Button style (similar to MenuStage)

String buttonStyle = "-fx-background-color: linear-gradient(to bottom right, #000428, #004e92);"

+ "-fx-text-fill: white;"

+ "-fx-font-size: 16px;"

+ "-fx-font-weight: bold;"

+ "-fx-padding: 15px 30px;"

+ "-fx-background-radius: 10;"

+ "-fx-pref-width: 300;";

// Buttons

Button addButton = new Button("ADD A NEW EMPLOYEE RECORD");

addButton.setStyle(buttonStyle);

addButton.setFont(Font.font("Arial", 14));

addButton.setOnAction(e -> handleAddEmployeeAction(employeeStage));

Button updateButton = new Button("UPDATE AN EMPLOYEE RECORD");

updateButton.setStyle(buttonStyle);

updateButton.setFont(Font.font("Arial", 14));

updateButton.setOnAction(e -> handleUpdateEmployeeAction(employeeStage));

Button deleteButton = new Button("DELETE AN EMPLOYEE RECORD");

deleteButton.setStyle(buttonStyle);

deleteButton.setFont(Font.font("Arial", 14));

deleteButton.setOnAction(e -> handleDeleteEmployeeAction(employeeStage));

Button viewLiveInButton = new Button("VIEW THE LIVE-IN EMPLOYEE DETAILS");

viewLiveInButton.setStyle(buttonStyle);

viewLiveInButton.setFont(Font.font("Arial", 14));

viewLiveInButton.setOnAction(e -> handleViewLiveInEmployees(employeeStage));

// Add components to the root layout

root.getChildren().addAll(topBar, addButton, updateButton, deleteButton, viewLiveInButton);

// Scene and stage setup

Scene scene = new Scene(root, 800, 600);

employeeStage.setScene(scene);

employeeStage.setTitle("Employee Menu");

employeeStage.show();

}

private void handleAddEmployeeAction(Stage employeeStage) {

employeeStage.close();

new AddEmployeeStage(employeeStage).show();

}

private void handleUpdateEmployeeAction(Stage employeeStage) {

employeeStage.close();

new UpdateEmployeeStage(employeeStage).show();

}

private void handleDeleteEmployeeAction(Stage employeeStage) {

employeeStage.close();

new DeleteEmployeeStage(employeeStage).show();

}

private void handleViewLiveInEmployees(Stage employeeStage) {

employeeStage.close();

new ViewLiveInEmployeesStage(employeeStage).show();

}

}

**AddEmployeeStage.java**

package com.example.project;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.VBox;

import javafx.scene.layout.HBox;

import javafx.stage.Stage;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

public class AddEmployeeStage {

private Stage menuStage; // Reference to the MenuStage

public AddEmployeeStage(Stage menuStage) {

this.menuStage = menuStage; // Save reference to menuStage passed from MenuStage

}

public void show() {

// Create a new stage for adding employee details

Stage addEmployeeStage = new Stage();

// Create a VBox layout to organize the form fields

VBox layout = new VBox(15); // 15px spacing between fields

layout.setStyle("-fx-background-color: linear-gradient(to bottom right, #0f0c29, #302b63, #24243e); -fx-padding: 20px;");

layout.setAlignment(Pos.CENTER);

// Define universal styles for the UI elements

String labelStyle = "-fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;";

String textFieldStyle = "-fx-background-color: white; -fx-border-radius: 8px; -fx-padding: 5px; -fx-font-size: 12px;";

String buttonStyle = "-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;";

// Create Labels and TextFields for employeeId, Name, Age, and CNIC

Label employeeIdLabel = new Label("Employee ID:");

employeeIdLabel.setStyle(labelStyle);

TextField employeeIdField = new TextField();

employeeIdField.setStyle(textFieldStyle);

Label nameLabel = new Label("Name:");

nameLabel.setStyle(labelStyle);

TextField nameField = new TextField();

nameField.setStyle(textFieldStyle);

Label ageLabel = new Label("Age:");

ageLabel.setStyle(labelStyle);

TextField ageField = new TextField();

ageField.setStyle(textFieldStyle);

Label cnicLabel = new Label("CNIC:");

cnicLabel.setStyle(labelStyle);

TextField cnicField = new TextField();

cnicField.setStyle(textFieldStyle);

// Create an "Add" button to save the employee details

Button addButton = new Button("Add");

addButton.setStyle(buttonStyle);

// Add hover effect for the "Add" button

addButton.setOnMouseEntered(e -> addButton.setStyle("-fx-background-color: #45a049; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;"));

addButton.setOnMouseExited(e -> addButton.setStyle(buttonStyle));

// Add event handler for the "Add" button

addButton.setOnAction(e -> {

String employeeId = employeeIdField.getText();

String name = nameField.getText();

String age = ageField.getText();

String cnic = cnicField.getText();

if (employeeId.isEmpty() || name.isEmpty() || age.isEmpty() || cnic.isEmpty()) {

showAlert("Error", "All fields are required", Alert.AlertType.ERROR);

} else {

// Write the employee's information to the Employees.txt file

try (BufferedWriter writer = new BufferedWriter(new FileWriter("Employees.txt", true))) {

// Create a string with employee details

String employeeData = employeeId + "," + name + "," + age + "," + cnic + System.lineSeparator();

// Write to the file

writer.write(employeeData);

showAlert("Success", "The employee has been added successfully.", Alert.AlertType.INFORMATION);

// Clear the fields for new employee entry, keeping the AddEmployeeStage open

employeeIdField.clear();

nameField.clear();

ageField.clear();

cnicField.clear();

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to add employee. Please try again.", Alert.AlertType.ERROR);

}

}

});

// Create a "Display" button to view employees

Button displayButton = new Button("Display");

displayButton.setStyle(buttonStyle);

// Add hover effect for the "Display" button

displayButton.setOnMouseEntered(e -> displayButton.setStyle("-fx-background-color: #45a049; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;"));

displayButton.setOnMouseExited(e -> displayButton.setStyle(buttonStyle));

// Add event handler for the "Display" button

displayButton.setOnAction(e -> {

addEmployeeStage.close(); // Close the AddEmployeeStage

ViewLiveInEmployeesStage viewStage = new ViewLiveInEmployeesStage(addEmployeeStage);

viewStage.show();

});

// Create an HBox to hold the Add and Display buttons

HBox buttonBox = new HBox(10); // 10px spacing between buttons

buttonBox.setAlignment(Pos.CENTER);

buttonBox.getChildren().addAll(addButton, displayButton);

// Back Button (Top Right Corner)

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #4569a0; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

backButton.setOnAction(e -> {

addEmployeeStage.close(); // Close the AddEmployeeStage

menuStage.show(); // Show the MenuStage again

});

// Create an HBox to position the Back button at the top-right

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_RIGHT); // Align the button to the top-right

topBar.getChildren().add(backButton); // Add the Back button to the top bar

// Add all components to the layout

layout.getChildren().addAll(topBar, employeeIdLabel, employeeIdField, nameLabel, nameField, ageLabel, ageField, cnicLabel, cnicField, buttonBox);

// Create and show the scene for the add employee stage

Scene scene = new Scene(layout, 600, 500); // Adjusted scene size

addEmployeeStage.setScene(scene);

addEmployeeStage.setTitle("Add New Employee");

addEmployeeStage.show();

}

private void showAlert(String title, String content, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(content);

alert.showAndWait();

}

}

**UpdateEmployeeStage.java**

package com.example.project;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.VBox;

import javafx.scene.layout.HBox;

import javafx.stage.Stage;

import java.io.\*;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.List;

public class UpdateEmployeeStage {

private Stage menuStage; // Reference to the MenuStage

public UpdateEmployeeStage(Stage menuStage) {

this.menuStage = menuStage; // Save reference to menuStage passed from MenuStage

}

public void show() {

// Create a new stage for updating employee details

Stage updateEmployeeStage = new Stage();

// Create a VBox layout to organize the form fields

VBox layout = new VBox(15); // 15px spacing between fields

layout.setStyle("-fx-background-color: linear-gradient(to bottom right, #0f0c29, #302b63, #24243e); -fx-padding: 20px;");

layout.setAlignment(Pos.CENTER);

// Define a universal label style

String labelStyle = "-fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;";

// Label and TextField to search by employee's ID

Label searchLabel = new Label("Enter Employee ID to Update:");

searchLabel.setStyle(labelStyle);

TextField searchField = new TextField();

searchField.setPromptText("Employee ID");

// Button to search the employee

Button searchButton = new Button("Search");

searchButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

// Create fields for displaying and updating employee details (now editable)

Label nameLabel = new Label("Name:");

nameLabel.setStyle(labelStyle);

TextField nameField = new TextField();

Label ageLabel = new Label("Age:");

ageLabel.setStyle(labelStyle);

TextField ageField = new TextField();

Label cnicLabel = new Label("CNIC:");

cnicLabel.setStyle(labelStyle);

TextField cnicField = new TextField();

// Create an "Update" button to save the updated employee details

Button updateButton = new Button("Update");

updateButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

updateButton.setDisable(true); // Initially disabled until the employee is found

// Create a "Display" button to view employee data

Button displayButton = new Button("Display");

displayButton.setStyle("-fx-background-color: #FFA500; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

displayButton.setOnAction(e -> {

updateEmployeeStage.close();

new ViewLiveInEmployeesStage(updateEmployeeStage).show();

});

// Event handler for search button

searchButton.setOnAction(e -> handleSearchAction(searchField, nameField, ageField, cnicField, updateButton));

// Event handler for update button

updateButton.setOnAction(e -> handleUpdateAction(searchField, nameField, ageField, cnicField, updateButton));

// Back Button (Top Right Corner)

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #4569a0; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

backButton.setOnAction(e -> {

updateEmployeeStage.close(); // Close the UpdateEmployeeStage

menuStage.show(); // Show the MenuStage again

});

// Create an HBox to position the Back button at the top-right

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_RIGHT); // Align the button to the top-right

topBar.getChildren().add(backButton); // Add the Back button to the top bar

// Create an HBox for the Update and Display buttons

HBox actionButtons = new HBox(10); // 10px spacing between buttons

actionButtons.setAlignment(Pos.CENTER);

actionButtons.getChildren().addAll(updateButton, displayButton);

// Add all components to the layout

layout.getChildren().addAll(topBar, searchLabel, searchField, searchButton,

nameLabel, nameField,

ageLabel, ageField,

cnicLabel, cnicField,

actionButtons);

// Create and show the scene for the update employee stage

Scene scene = new Scene(layout, 600, 600);

updateEmployeeStage.setScene(scene);

updateEmployeeStage.setTitle("Update Employee Record");

updateEmployeeStage.show();

}

private void handleSearchAction(TextField searchField, TextField nameField, TextField ageField, TextField cnicField, Button updateButton) {

String searchID = searchField.getText().trim();

if (searchID.isEmpty()) {

showAlert("Error", "Employee ID is required", "Please enter an employee ID to search.", Alert.AlertType.ERROR);

return;

}

try (BufferedReader reader = new BufferedReader(new FileReader("Employees.txt"))) {

String line;

boolean found = false;

while ((line = reader.readLine()) != null) {

String[] employeeData = line.split(",");

if (employeeData[0].equals(searchID)) {

// Populate the fields with existing data

nameField.setText(employeeData[1]);

ageField.setText(employeeData[2]);

cnicField.setText(employeeData[3]);

updateButton.setDisable(false); // Enable the update button

found = true;

break;

}

}

if (!found) {

showAlert("Employee Not Found", "Employee Not Found", "No employee found with the ID: " + searchID, Alert.AlertType.ERROR);

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to Search", "There was an error searching for the employee.", Alert.AlertType.ERROR);

}

}

private void handleUpdateAction(TextField searchField, TextField nameField, TextField ageField, TextField cnicField, Button updateButton) {

String searchID = searchField.getText().trim();

String newName = nameField.getText().trim();

String age = ageField.getText().trim();

String cnic = cnicField.getText().trim();

if (newName.isEmpty() || age.isEmpty() || cnic.isEmpty()) {

showAlert("Error", "All fields are required", "Please fill in all the fields.", Alert.AlertType.ERROR);

return;

}

try {

List<String> lines = Files.readAllLines(Paths.get("Employees.txt"));

boolean updated = false;

for (int i = 0; i < lines.size(); i++) {

String[] employeeData = lines.get(i).split(",");

if (employeeData[0].equals(searchID)) {

lines.set(i, searchID + "," + newName + "," + age + "," + cnic);

updated = true;

break;

}

}

if (updated) {

Files.write(Paths.get("Employees.txt"), lines);

showAlert("Success", "Employee Updated", "The employee's record has been updated successfully.", Alert.AlertType.INFORMATION);

// Clear the fields and keep the stage open for further updates

nameField.clear();

ageField.clear();

cnicField.clear();

// Optionally, disable the update button again after update

updateButton.setDisable(true);

} else {

showAlert("Error", "Update Failed", "No matching employee found to update.", Alert.AlertType.ERROR);

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to Update", "There was an error updating the employee record.", Alert.AlertType.ERROR);

}

}

private void showAlert(String title, String header, String content, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(header);

alert.setContentText(content);

alert.showAndWait();

}

}

**DeleteEmployeeStage.java**

package com.example.project;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.io.\*;

import java.util.ArrayList;

import java.util.List;

public class DeleteEmployeeStage {

private Stage menuStage; // Reference to MenuStage

public DeleteEmployeeStage(Stage menuStage) {

this.menuStage = menuStage; // Save reference to menuStage passed from MenuStage

}

public void show() {

// Create a new stage for deleting an employee

Stage deleteEmployeeStage = new Stage();

deleteEmployeeStage.setTitle("Delete Employee");

// Create a VBox layout for the delete employee form

VBox layout = new VBox(15); // 15px spacing between fields

layout.setStyle("-fx-background-color: linear-gradient(to bottom right, #0f0c29, #302b63, #24243e); -fx-padding: 20px;");

layout.setAlignment(Pos.CENTER);

// Define style for the UI elements

String labelStyle = "-fx-text-fill: white; -fx-font-size: 14px; -fx-font-weight: bold;";

String textFieldStyle = "-fx-background-color: white; -fx-border-radius: 10px; -fx-padding: 10px; -fx-font-size: 14px;";

String buttonStyle = "-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;";

// Create a Label and TextField for employee ID input

Label employeeIdLabel = new Label("Enter Employee ID to Delete:");

employeeIdLabel.setStyle(labelStyle);

TextField employeeIdField = new TextField();

employeeIdField.setStyle(textFieldStyle);

// Create Labels to display employee details (initially hidden)

Label nameLabel = new Label("Name: ");

nameLabel.setStyle(labelStyle);

Label ageLabel = new Label("Age: ");

ageLabel.setStyle(labelStyle);

Label cnicLabel = new Label("CNIC: ");

cnicLabel.setStyle(labelStyle);

// Initially disable the delete button

Button deleteButton = new Button("Delete");

deleteButton.setStyle(buttonStyle);

deleteButton.setDisable(true);

// Back Button (Top Right Corner)

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #4569a0; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

backButton.setOnAction(e -> {

deleteEmployeeStage.close(); // Close the DeleteEmployeeStage

menuStage.show(); // Show the MenuStage again

});

// Add event handler for the "Search" button

Button searchButton = new Button("Search");

searchButton.setStyle(buttonStyle);

searchButton.setOnAction(e -> handleSearchAction(employeeIdField, nameLabel, ageLabel, cnicLabel, deleteButton));

// Add event handler for the "Delete" button

deleteButton.setOnAction(e -> handleDeleteAction(employeeIdField, deleteButton, nameLabel, ageLabel, cnicLabel));

// Add Display Button

Button displayButton = new Button("Display");

displayButton.setStyle(buttonStyle);

displayButton.setOnAction(e -> {

deleteEmployeeStage.close();

new ViewLiveInEmployeesStage(deleteEmployeeStage).show();

});

// Create an HBox for Delete and Display buttons

HBox buttonBox = new HBox(10); // 10px spacing

buttonBox.setAlignment(Pos.CENTER);

buttonBox.getChildren().addAll(deleteButton, displayButton);

// Create an HBox to position the Back button at the top-right

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_RIGHT); // Align the button to the top-right

topBar.getChildren().add(backButton); // Add the Back button to the top bar

// Add the input field, search button, labels, and buttons to the layout

layout.getChildren().addAll(topBar, employeeIdLabel, employeeIdField, searchButton, nameLabel, ageLabel, cnicLabel, buttonBox);

// Create and show the scene for the delete employee stage

Scene scene = new Scene(layout, 500, 500);

deleteEmployeeStage.setScene(scene);

deleteEmployeeStage.show();

}

private void handleSearchAction(TextField employeeIdField, Label nameLabel, Label ageLabel, Label cnicLabel, Button deleteButton) {

String employeeId = employeeIdField.getText().trim();

if (employeeId.isEmpty()) {

showAlert("Error", "Employee ID is required", "Please enter the ID of the employee to search.", Alert.AlertType.ERROR);

return;

}

// Search for employee in the file

File employeeFile = new File("Employees.txt");

boolean employeeFound = false;

try (BufferedReader reader = new BufferedReader(new FileReader(employeeFile))) {

String line;

while ((line = reader.readLine()) != null) {

String[] employeeData = line.split(",");

if (employeeData.length >= 4 && employeeData[0].equals(employeeId)) {

// Display employee details

nameLabel.setText("Name: " + employeeData[1]);

ageLabel.setText("Age: " + employeeData[2]);

cnicLabel.setText("CNIC: " + employeeData[3]);

// Enable delete button

deleteButton.setDisable(false);

employeeFound = true;

break;

}

}

if (!employeeFound) {

showAlert("Employee Not Found", null, "No employee found with the ID: " + employeeId, Alert.AlertType.ERROR);

deleteButton.setDisable(true);

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to Search", "There was an error reading the employee data.", Alert.AlertType.ERROR);

}

}

private void handleDeleteAction(TextField employeeIdField, Button deleteButton, Label nameLabel, Label ageLabel, Label cnicLabel) {

String employeeId = employeeIdField.getText().trim();

if (employeeId.isEmpty()) {

showAlert("Error", "Employee ID is required", "Please enter the ID of the employee to delete.", Alert.AlertType.ERROR);

return;

}

// Search for employee in the file and delete the record

File employeeFile = new File("Employees.txt");

List<String> updatedContent = new ArrayList<>();

boolean employeeFound = false;

try (BufferedReader reader = new BufferedReader(new FileReader(employeeFile))) {

String line;

while ((line = reader.readLine()) != null) {

String[] employeeData = line.split(",");

if (employeeData.length >= 4 && employeeData[0].equals(employeeId)) {

employeeFound = true; // Employee found, skipping the record to delete it

continue;

}

updatedContent.add(line); // Add other employees to the updated list

}

if (!employeeFound) {

showAlert("Employee Not Found", null, "No employee found with the ID: " + employeeId, Alert.AlertType.ERROR);

} else {

// Overwrite the file with the updated content (without the deleted employee)

try (BufferedWriter writer = new BufferedWriter(new FileWriter(employeeFile))) {

for (String lineContent : updatedContent) {

writer.write(lineContent);

writer.newLine();

}

}

showAlert("Success", "Employee Deleted", "The employee record has been successfully deleted.", Alert.AlertType.INFORMATION);

// Reset the labels and disable the delete button

nameLabel.setText("Name: ");

ageLabel.setText("Age: ");

cnicLabel.setText("CNIC: ");

employeeIdField.clear();

deleteButton.setDisable(true);

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to Delete Employee", "There was an error reading or writing the employee data.", Alert.AlertType.ERROR);

}

}

private void showAlert(String title, String header, String content, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(header);

alert.setContentText(content);

alert.showAndWait();

}

}

**ViewLiveInEmployeesStage.java**

package com.example.project;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.VBox;

import javafx.scene.layout.HBox;

import javafx.stage.Stage;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class ViewLiveInEmployeesStage {

private Stage previousStage; // Reference to the previous stage

public ViewLiveInEmployeesStage(Stage previousStage) {

this.previousStage = previousStage; // Save reference to the previous stage passed from MenuStage

}

public void show() {

// Create a new stage to display the live-in employees

Stage viewStage = new Stage();

viewStage.setTitle("Live-In Employees");

// Create a VBox layout to hold the TableView

VBox layout = new VBox(10);

layout.setStyle("-fx-background-color: linear-gradient(to bottom right, #0f0c29, #302b63, #24243e); -fx-padding: 20px;");

layout.setAlignment(Pos.CENTER);

// Create the TableView and its columns

TableView<Employee> tableView = new TableView<>();

// Create the columns for the TableView

TableColumn<Employee, String> idColumn = new TableColumn<>("Employee ID");

idColumn.setCellValueFactory(new PropertyValueFactory<>("employeeId"));

TableColumn<Employee, String> nameColumn = new TableColumn<>("Name");

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

TableColumn<Employee, String> ageColumn = new TableColumn<>("Age");

ageColumn.setCellValueFactory(new PropertyValueFactory<>("age"));

TableColumn<Employee, String> cnicColumn = new TableColumn<>("CNIC");

cnicColumn.setCellValueFactory(new PropertyValueFactory<>("cnic"));

// Add columns to the TableView

tableView.getColumns().addAll(idColumn, nameColumn, ageColumn, cnicColumn);

// Read the data from the Employees.txt file and populate the TableView

try (BufferedReader reader = new BufferedReader(new FileReader("Employees.txt"))) {

String line;

while ((line = reader.readLine()) != null) {

String[] employeeData = line.split(",");

if (employeeData.length == 4) { // Ensure we have 4 fields (ID, Name, Age, CNIC)

// Create a new Employee object and add it to the TableView

Employee employee = new Employee(employeeData[0], employeeData[1], employeeData[2], employeeData[3]);

tableView.getItems().add(employee);

}

}

} catch (IOException ex) {

ex.printStackTrace();

showAlert("Error", "Failed to Load Employees", "There was an error reading the employee data.", Alert.AlertType.ERROR);

}

// Create a Back button (Top Right Corner)

Button backButton = new Button("Back");

backButton.setStyle("-fx-background-color: #4569a0; -fx-text-fill: white; -fx-font-size: 16px; -fx-padding: 12px 20px; -fx-border-radius: 8px;");

backButton.setOnAction(e -> {

viewStage.close(); // Close the ViewLiveInEmployeesStage

previousStage.show(); // Show the previous stage (MenuStage or AddEmployeeStage)

});

// Create an HBox to position the Back button at the top-right

HBox topBar = new HBox();

topBar.setAlignment(Pos.TOP\_RIGHT); // Align the button to the top-right

topBar.getChildren().add(backButton); // Add the Back button to the top bar

// Add all components to the layout

layout.getChildren().addAll(topBar, tableView);

// Create and show the scene for the view employees stage

Scene scene = new Scene(layout, 800, 600);

viewStage.setScene(scene);

viewStage.show();

}

private void showAlert(String title, String header, String content, Alert.AlertType alertType) {

Alert alert = new Alert(alertType);

alert.setTitle(title);

alert.setHeaderText(header);

alert.setContentText(content);

alert.showAndWait();

}

}