Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EmployeeProject

{

internal class Program

{

static void Main(string[] args)

{

Program program = new Program();

program.manageMenu();

Console.ReadKey();

}

void manageMenu()

{

Console.WriteLine("Welcome to Employee menu management");

int choice = 0;

ManageMenu menu = new ManageMenu();

do

{

Console.WriteLine("1: Add Employee");

Console.WriteLine("2: Edit Employee Age");

Console.WriteLine("3: Remove Employee");

Console.WriteLine("4: Print A Employee Details");

Console.WriteLine("5: Print All Employee Details");

Console.WriteLine("0: Exit");

while (!int.TryParse(Console.ReadLine(), out choice))

{

Console.WriteLine("Try again. Please enter a number");

}

try

{

switch (choice)

{

case 1:

menu.AddEmployees();

break;

case 2:

menu.EditEmployeeAge();

break;

case 3:

menu.RemoveEmployee();

break;

case 4:

menu.PrintSingleEmployeeByID();

break;

case 5:

menu.PrintEmployees();

break;

case 0:

Console.WriteLine("Bye bye..........");

break;

default:

Console.WriteLine("Invalid choice. Please try again");

break;

}

}

catch (NullReferenceException nre)

{

Console.WriteLine("Null by mistake");

Console.WriteLine(nre.Message);

}

catch (ArgumentOutOfRangeException aore)

{

Console.WriteLine("The employee could not be found");

Console.WriteLine(aore.Message);

}

catch (FormatException fe)

{

Console.WriteLine("expecting a number");

Console.WriteLine(fe.Message);

}

catch (Exception e)

{

Console.WriteLine("Something went wrong");

Console.WriteLine(e.Message);

}

} while (choice != 0);

}

}

}

ManageMenu.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using EmployeeDALLibrary;

using EmployeeModelsLibrary;

namespace EmployeeProject

{

internal class ManageMenu

{

List<Employee> employees;

EmployeeDAL employeeDAL;

public ManageMenu()

{

employeeDAL = new EmployeeDAL();

}

void GetAllEmployees()

{

employees = null;

try

{

employees = employeeDAL.GetAllEmployees().ToList();

}

catch (NoEmployeeException npe)

{

Console.WriteLine(npe.Message);

}

catch (Exception npe)

{

Console.WriteLine("Something went wrong. Will fix soon...");

Console.WriteLine(npe.Message);

}

}

public void AddEmployees()

{

Employee employee = new Employee();

employee.GetEmployeeDetails();

try

{

employeeDAL.InsertNewEmployee(employee);

}

catch (Exception e)

{

Console.WriteLine("Could not add the employee");

Console.WriteLine(e.Message);

}

}

public void EditEmployeeAge()

{

int id = GetIdFromUser();

Employee employee = GetEmployeeById(id);

if (employee == null)

{

Console.WriteLine("Invalid Employee Id. Cannot edit");

return;

}

Console.WriteLine("The employee for you have chosen to edit age");

PrintEmployee(employee);

int age;

Console.WriteLine("Please enter the new age");

while (!int.TryParse(Console.ReadLine(), out age))

{

Console.WriteLine("Invalid input for age. Please try again...");

}

employee.emp\_age = age;

if (employeeDAL.UpdateEmployeeAge (id, ((int)age)))

Console.WriteLine("Updated. New Details");

PrintEmployee(employee);

}

public void RemoveEmployee()

{

int id = GetIdFromUser();

int idx = -1;

try

{

idx = employees.SingleOrDefault(e => e.emp\_id == id).emp\_id;

}

catch (Exception)

{

Console.WriteLine("No such employee");

}

Employee employee = GetEmployeeById(id);

if (idx != -1)

{

Console.WriteLine("Do you want to delete the following employee???");

PrintEmployee(employee);

string check = Console.ReadLine();

if (check == "yes")

{

if (employeeDAL.RemoveEmployee(id))

Console.WriteLine("Removed successfully");

employees.RemoveAt(idx);

}

}

}

public void PrintSingleEmployeeByID()

{

int id = GetIdFromUser();

Employee employee = GetEmployeeById(id);

if (employee != null)

{

PrintEmployee(employee);

}

else

Console.WriteLine("No such pizza");

}

public void PrintEmployees()

{

GetAllEmployees();

var sortedPizzas = employees.OrderBy(e => e.emp\_id);

foreach (var item in sortedPizzas)

{

if (item != null)

PrintEmployee(item);

}

}

int GetIdFromUser()

{

Console.WriteLine("Please enter the employee id");

int id;

while (!int.TryParse(Console.ReadLine(), out id))

{

Console.WriteLine("Invalid entry Employee ID. Please try again...");

}

return id;

}

public Employee GetEmployeeById(int id)

{

GetAllEmployees();

Employee employee = employees.SingleOrDefault(e => e.emp\_id == id);

return employee;

}

private void PrintEmployee(Employee item)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine(item);

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

}

}

EmployeeDAL.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using EmployeeModelsLibrary;

using System.Data;

namespace EmployeeDALLibrary

{

public class EmployeeDAL

{

SqlConnection conn;

public EmployeeDAL()

{

conn = MyConnection.GetConnection();

}

public ICollection<Employee> GetAllEmployees()

{

List<Employee> employees = new List<Employee>();

DataSet ds = new DataSet();

SqlDataAdapter adapter = new SqlDataAdapter("proc\_GetAllEmployees", conn);

adapter.SelectCommand.CommandType = CommandType.StoredProcedure;

adapter.Fill(ds);

Employee employee;

if (ds.Tables[0].Rows.Count == 0)

throw new NoEmployeeException();

foreach (DataRow item in ds.Tables[0].Rows)

{

employee = new Employee();

employee.emp\_id = Convert.ToInt32(item[0]);

employee.emp\_name = item[1].ToString();

employee.emp\_age = Convert.ToInt32(item[2]);

employees.Add(employee);

}

return employees;

}

public bool InsertNewEmployee(Employee employee)

{

SqlCommand cmd = new SqlCommand("proc\_InsertEmployee", conn);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@emp\_name", employee.emp\_name);

cmd.Parameters.AddWithValue("@emp\_age", employee.emp\_age);

conn.Open();

if (cmd.ExecuteNonQuery() > 0)

{

conn.Close();

return true;

}

conn.Close();

return false;

}

public bool UpdateEmployeeAge(int id, int age)

{

SqlCommand cmd = new SqlCommand("proc\_UpdateEmployeeAge", conn);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@emp\_id", id);

cmd.Parameters.AddWithValue("@emp\_age", age);

conn.Open();

if (cmd.ExecuteNonQuery() > 0)

{

conn.Close();

return true;

}

conn.Close();

return false;

}

public bool RemoveEmployee(int id)

{

SqlCommand cmd = new SqlCommand("proc\_RemoveEmployee", conn);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@emp\_id", id);

conn.Open();

if (cmd.ExecuteNonQuery() > 0)

{

conn.Close();

return true;

}

conn.Close();

return false;

}

}

}

Employee.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EmployeeModelsLibrary

{

public class Employee

{

public int emp\_id { get; set; }

public string emp\_name { get; set; }

public int emp\_age { get; set; }

public void GetEmployeeDetails()

{

Console.WriteLine("Please enter the employee name");

emp\_name = Console.ReadLine();

Console.WriteLine("Please enter the employee's age");

int age;

while (!int.TryParse(Console.ReadLine(), out age))

{

Console.WriteLine("Invalid entry for age. Please try again...");

}

emp\_age = age;

}

public override string ToString()

{

return "Employee ID " + emp\_id

+ "\nEmployee Name " + emp\_name

+ "\nEmployee Age " + emp\_age;

}

}

}