QUEZON CITY UNIVERSITY

COLLEGE OF COMPUTER STUDIES
INFORMATION TECHNOLOGY

IPT102 – INTEGRATIVE PROGRAMMING AND TECHNOLOGIES

WEEK 14 LABORATORY ACTIVITY

NAME: CAHILIG, EFREN DAVE G. and LUMAGUE, JHAYRON U.

STUDENT NO: 23-2396 and 23-2335

YEAR/SECTION: SBIT-3R

DATE: 11/18/2025

SCORE	PERCENTAGE

IPT102 – Week 14: Entity Framework in ASP.NET MVC (Parts I & II)
covering Database Creation, Entity Framework Setup, and CRUD Operations.

Laboratory Activity
Title: Implementing Entity Framework in ASP.NET MVC using Database-First Approach
Course: IPT102 – Integrative Programming and Technologies II
Software: Visual Studio 2022, SQL Server Management Studio (SSMS)
Framework: .NET Framework 4.7.2
Approach: Database-First using Entity Framework 6.x

Learning Objectives
After completing this activity, students should be able to:

1. **Create a database and table in SQL Server.**
2. **Integrate Entity Framework Database-First model in an MVC project.**
3. **Perform CRUD (Create, Read, Update, Delete) operations using Entity Framework.**
4. **Display data dynamically in ASP.NET MVC views.**

Instructions
STEP 1: Create the Database and Table

1. **Open SQL Server Management Studio (SSMS).**
2. **Execute the following SQL Script:**

```sql
CREATE DATABASE MVC_DB;
GO

USE MVC_DB;
GO

CREATE TABLE Employees (
    Id INT PRIMARY KEY IDENTITY(1,1),
    FullName NVARCHAR(100),
    Position NVARCHAR(50),
    Department NVARCHAR(50),
    Salary DECIMAL(10,2)
);

INSERT INTO Employees (FullName, Position, Department, Salary)
VALUES
('Juan Dela Cruz', 'Manager', 'Sales', 50000),
('Maria Santos', 'HR Officer', 'Human Resources', 35000),
('Jose Rizal', 'Developer', 'IT', 45000);
```

3. **Verify that the data has been inserted successfully.**

STEP 2: Create an ASP.NET MVC Project

1. **Open Visual Studio 2022 → File → New → Project.**
2. **Choose ASP.NET Web Application (.NET Framework).**
3. **Name your project:**
   EmployeeCRUD_MVC
4. **Select Empty Project, then check MVC in the Add Folders and Core References section.**
5. **Set Authentication to No Authentication.**

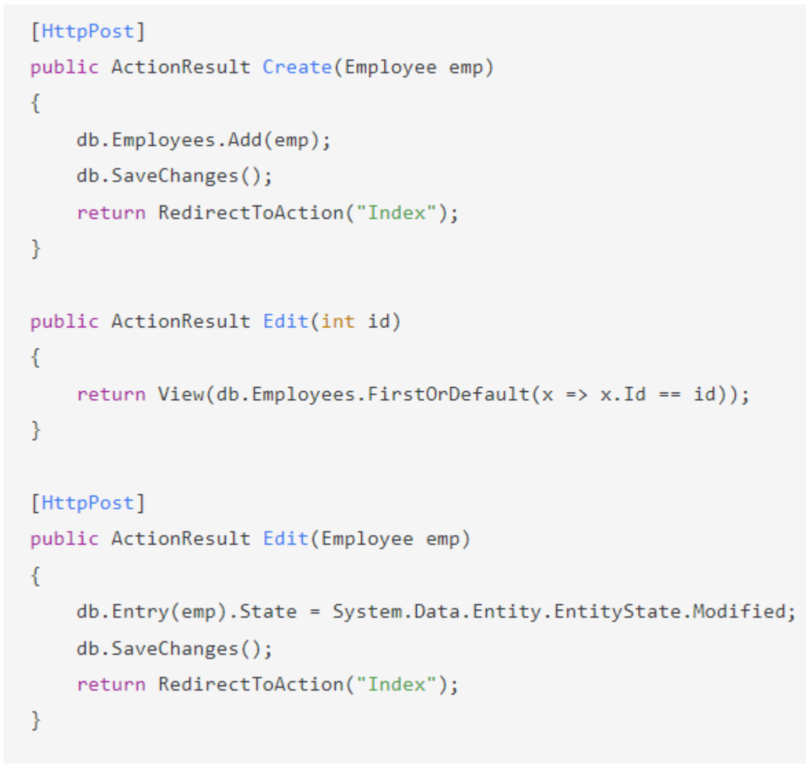
STEP 3: Add Entity Framework Model

1. **Right-click on Models Folder → Add → New Item.**
2. **Select ADO.NET Entity Data Model, name it:**
   EmployeeDataModel.edmx
3. **Choose EF Designer from Database → Click Next.**
4. **Create a new connection:**
   - **Server name: (your SQL Server instance)**
   - **Database name: MVC_DB**
5. **Name the connection string:**
   EmployeeDBContext
6. **Select Entity Framework 6.x and include the Employees table.**
7. **Click Finish.**
   The EDMX model, Employee class, and DBContext will be generated automatically.

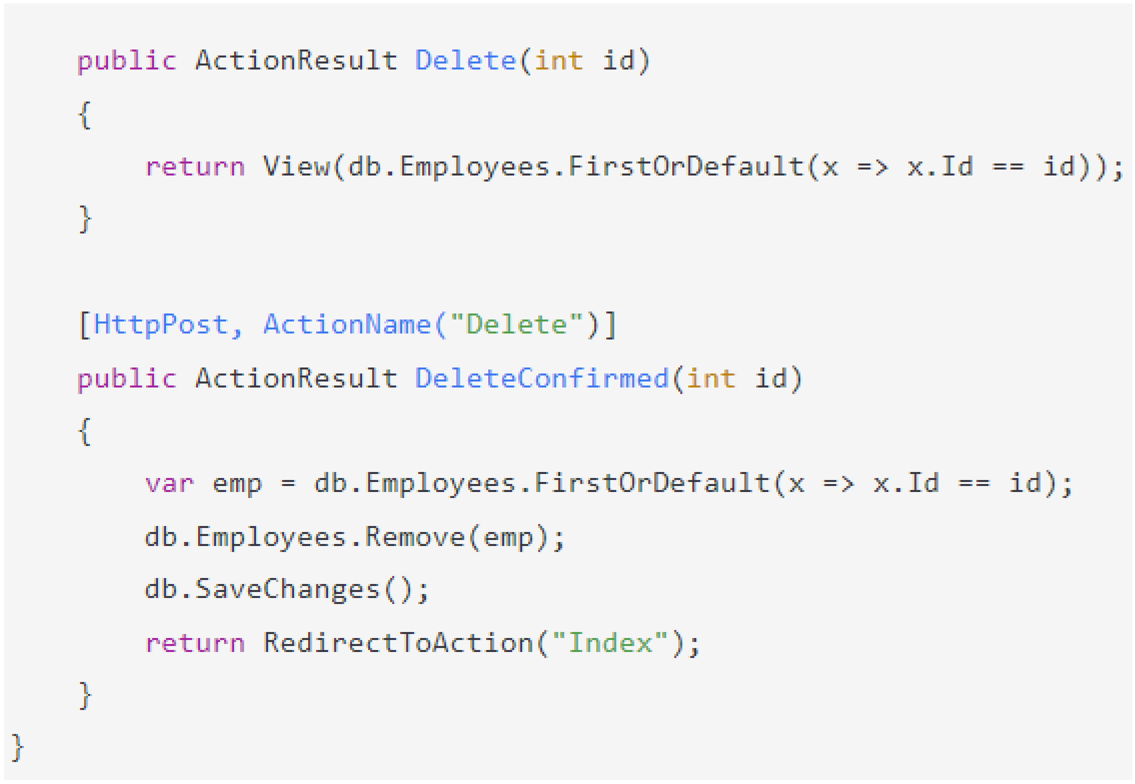
STEP 4: Create the Controller

1. **Right-click the Controllers Folder → Add → Controller.**
2. **Choose MVC 5 Controller – Empty.**
3. **Name it:**
   EmployeeController
4. **Modify the EmployeeController.cs file:**

```
using System.Linq;
using System.Web.Mvc;
using EmployeeCRUD_MVC.Models;

public class EmployeeController : Controller
{
    EmployeeDBContext db = new EmployeeDBContext();

    public ActionResult Index()
    {
        return View(db.Employees.ToList());
    }

    public ActionResult Details(int id)
    {
        return View(db.Employees.FirstOrDefault(x => x.Id == id));
    }

    public ActionResult Create()
    {
        return View();
    }
```



```
[HttpPost]
public ActionResult Create(Employee emp)
{
    db.Employees.Add(emp);
    db.SaveChanges();
    return RedirectToAction("Index");
}

public ActionResult Edit(int id)
{
    return View(db.Employees.FirstOrDefault(x => x.Id == id));
}

[HttpPost]
public ActionResult Edit(Employee emp)
{
    db.Entry(emp).State = System.Data.Entity.EntityState.Modified;
    db.SaveChanges();
    return RedirectToAction("Index");
}
```



```
    public ActionResult Delete(int id)
    {
        return View(db.Employees.FirstOrDefault(x => x.Id == id));
    }

    [HttpPost, ActionName("Delete")]
    public ActionResult DeleteConfirmed(int id)
    {
        var emp = db.Employees.FirstOrDefault(x => x.Id == id);
        db.Employees.Remove(emp);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
}
```

STEP 5: Create Views for CRUD Operations

1. **Right-click each Action Method → Add View.**
2. **For Index.cshtml, insert:**

```
@model IEnumerable<EmployeeCRUD_MVC.Models.Employee>
<h2>Employee List</h2>
<p><a href="@Url.Action("Create")">Add New Employee</a></p>
<table border="1">
    <tr>
        <th>ID</th>
        <th>Full Name</th>
        <th>Position</th>
        <th>Department</th>
        <th>Salary</th>
        <th>Actions</th>
    </tr>
    @foreach (var emp in Model)
    {
        <tr>
            <td>@emp.Id</td>
            <td>@emp.FullName</td>
            <td>@emp.Position</td>
            <td>@emp.Department</td>
            <td>@emp.Salary</td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id = emp.Id }) |
                @Html.ActionLink("Details", "Details", new { id = emp.Id }) |
                @Html.ActionLink("Delete", "Delete", new { id = emp.Id })
            </td>
        </tr>
    }
</table>
```

3. **Repeat for Create, Edit, Delete, and Details (use scaffolding).**

STEP 6: Modify Route Configuration
In App_Start/RouteConfig.cs, set:

```
defaults: new { controller = "Employee", action = "Index", id = UrlParameter.Optional }
```

Challenge Activity 1.
Add a Search Feature to the Employee Index View:

- **Add a textbox and button to search by FullName.**
- **Filter the displayed list using LINQ.**

```
public ActionResult Index(string search)
{
    var employees = from e in db.Employees select e;
    if (!string.IsNullOrEmpty(search))
        employees = employees.Where(e => e.FullName.Contains(search));
    return View(employees.ToList());
}
```

Submission Requirements

1. **Screenshots of the following:**
   - **SQL Database with test data**
   - **Entity Data Model (.edmx) in Visual Studio**
   - **Running Index Page showing data**
   - **Create, Edit, and Delete pages in action**
2. **Submit a ZIP file of your project folder.**

Challenge Activities 2

Challenge 1: Search Functionality

Add a **Search Feature** to the Employee Index View:

- Add a textbox and button to search employees by FullName.
- Use LINQ to filter results dynamically.

public ActionResult Index(string search)

```
public ActionResult Index(string search)
{
    var employees = from e in db.Employees select e;
    if (!string.IsNullOrEmpty(search))
        employees = employees.Where(e => e.FullName.Contains(search));
    return View(employees.ToList());
}
```

Challenge 2: Sort by Salary or Department

Enhance the Employee list by allowing users to **sort** the displayed data:

- Add clickable links or dropdown options to sort employees by **Salary** (ascending/descending) or **Department**.
- Update the Index action method to handle sorting.

Example code:

```
public ActionResult Index(string sortOrder)
{
    ViewBag.SalarySortParam = String.IsNullOrEmpty(sortOrder) ? "salary_desc" : "";
    ViewBag.DeptSortParam = sortOrder == "Department" ? "dept_desc" : "Department";

    var employees = from e in db.Employees select e;

    switch (sortOrder)
    {
        case "salary_desc":
            employees = employees.OrderByDescending(e => e.Salary);
            break;
        case "Department":
            employees = employees.OrderBy(e => e.Department);
            break;
        case "dept_desc":
            employees = employees.OrderByDescending(e => e.Department);
            break;
        default:
            employees = employees.OrderBy(e => e.Salary);
            break;
    }
    return View(employees.ToList());
}
```

Then, in your **Index.cshtml**, add sorting links:

```
<p>
    Sort by:
    @Html.ActionLink("Salary", "Index", new { sortOrder = ViewBag.SalarySortParam }) |
    @Html.ActionLink("Department", "Index", new { sortOrder = ViewBag.DeptSortParam })
</p>
```

Challenge 3: Add Data Validation
Implement **Model Validation** to ensure users cannot leave fields empty or enter invalid data:

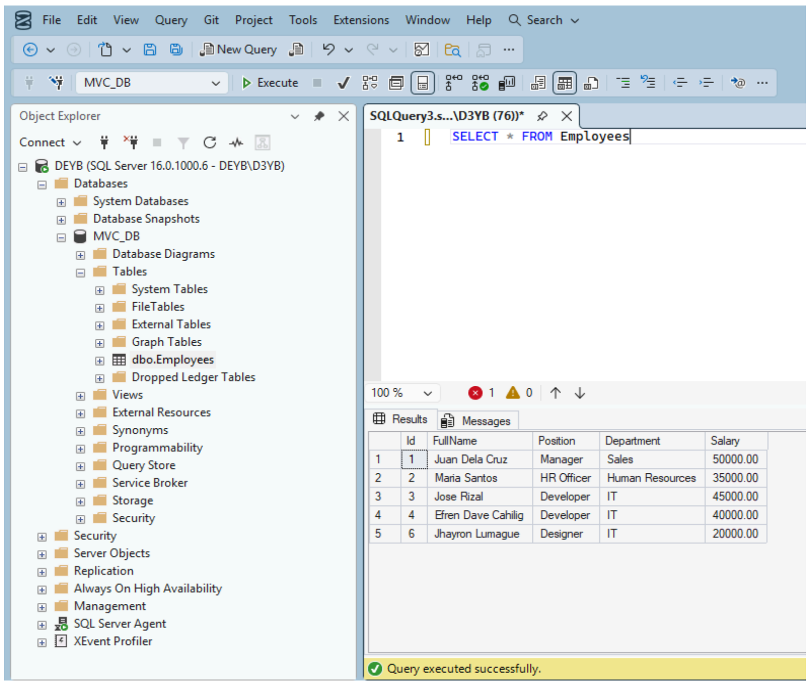
- Use Data Annotations in your Employee model.

```
public class Employee
{
    public int Id { get; set; }

    [Required(ErrorMessage = "Full Name is required")]
    public string FullName { get; set; }

    [Required]
    public string Position { get; set; }

    [Required]
    public string Department { get; set; }

    [Range(1000, 100000, ErrorMessage = "Salary must be between 1000 and 100000")]
    public decimal Salary { get; set; }
}
```

✅ **Expected Output:**
If a user submits an incomplete form, error messages should appear below each field.

SCREENSHOTS:

SAMPLE DATA :

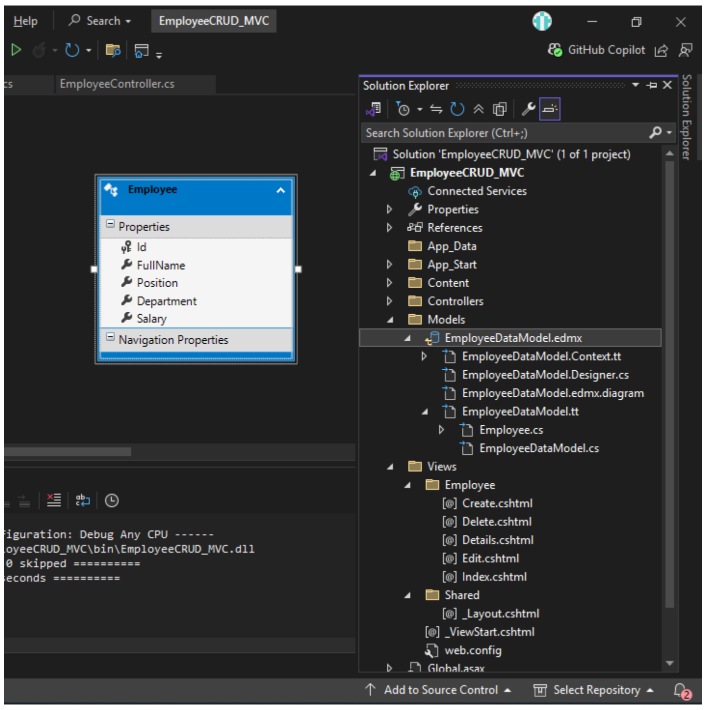


```
SELECT * FROM Employees
```

	Id	FullName	Position	Department	Salary
1	1	Juan Dela Cruz	Manager	Sales	50000.00
2	2	Maria Santos	HR Officer	Human Resources	35000.00
3	3	Jose Rizal	Developer	IT	45000.00
4	4	Efren Dave Cahilig	Developer	IT	40000.00
5	6	Jhayron Lumague	Designer	IT	20000.00

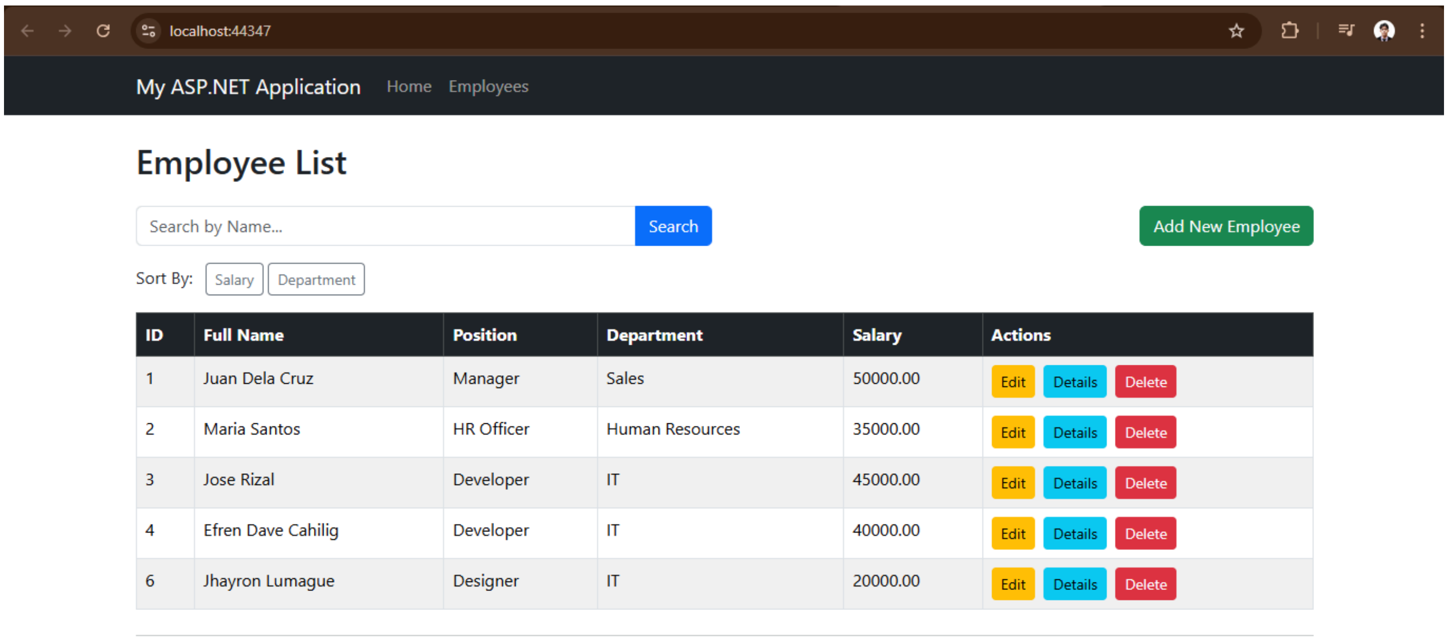
Query executed successfully.

ENTITY DATA MODEL:



```
figuration: Debug Any CPU ------
oyeeCRUD_MVC\bin\EmployeeCRUD_MVC.dll
0 skipped ==========
econds ==========
```

INDEX PAGE:



localhost:44347

My ASP.NET Application Home Employees

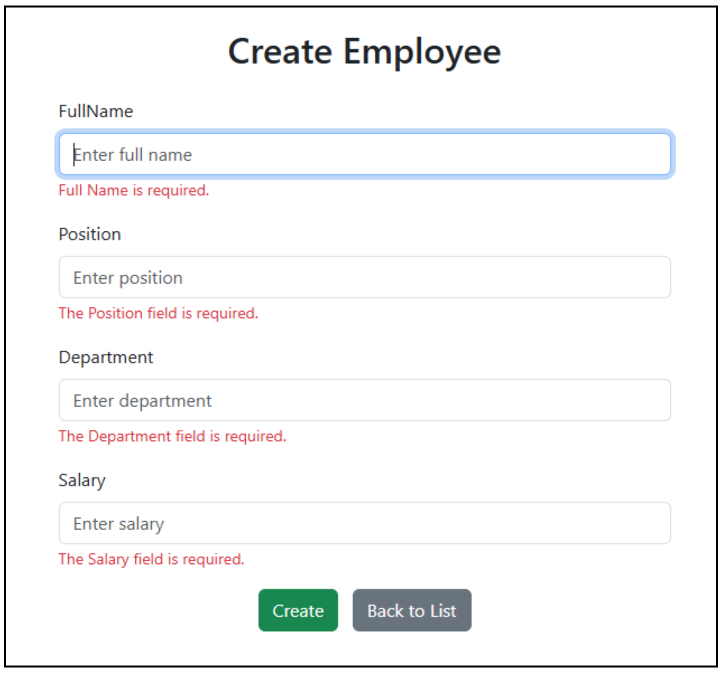
Employee List

Search by Name... Search | Add New Employee

Sort By: Salary Department

ID	Full Name	Position	Department	Salary	Actions
1	Juan Dela Cruz	Manager	Sales	50000.00	Edit Details Delete
2	Maria Santos	HR Officer	Human Resources	35000.00	Edit Details Delete
3	Jose Rizal	Developer	IT	45000.00	Edit Details Delete
4	Efren Dave Cahilig	Developer	IT	40000.00	Edit Details Delete
6	Jhayron Lumague	Designer	IT	20000.00	Edit Details Delete

CREATE (WITH VALIDATION):



Create Employee

FullName
Enter full name
Full Name is required.

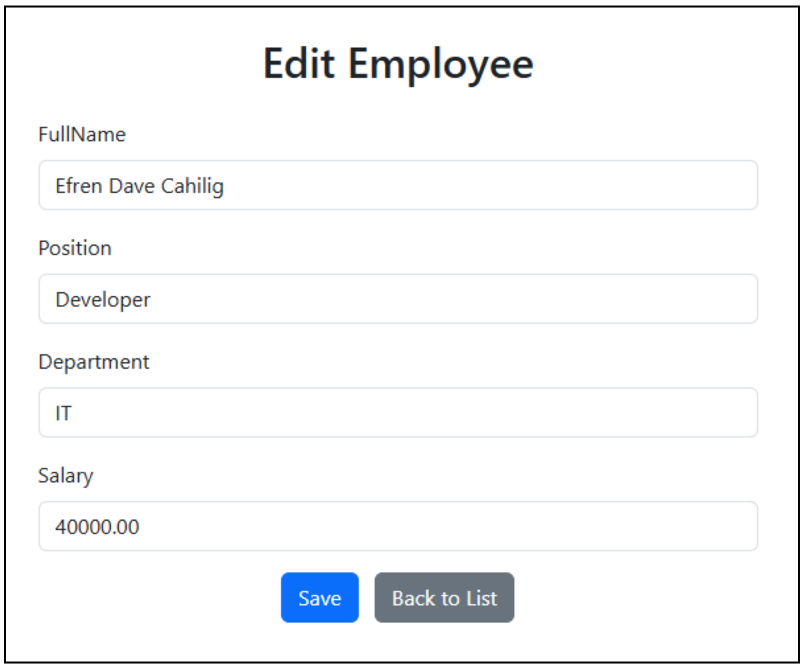
Position
Enter position
The Position field is required.

Department
Enter department
The Department field is required.

Salary
Enter salary
The Salary field is required.

Create Back to List

EDIT:



Edit Employee

FullName
Efren Dave Cahilig

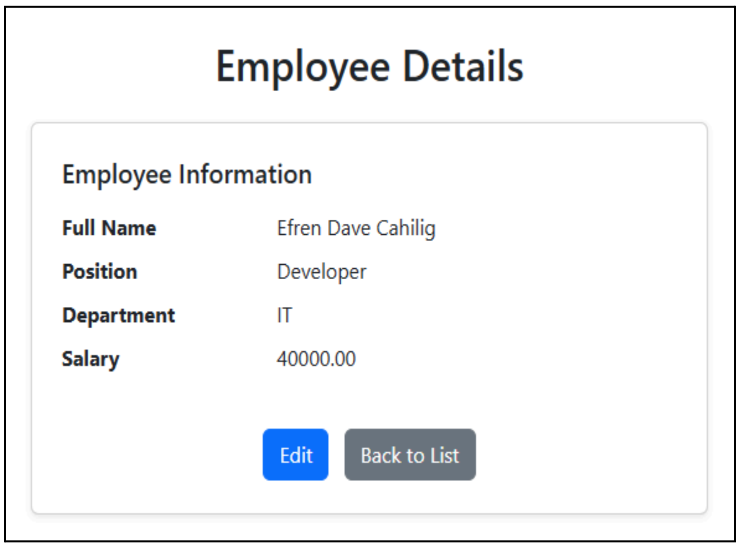
Position
Developer

Department
IT

Salary
40000.00

Save Back to List

DETAILS:



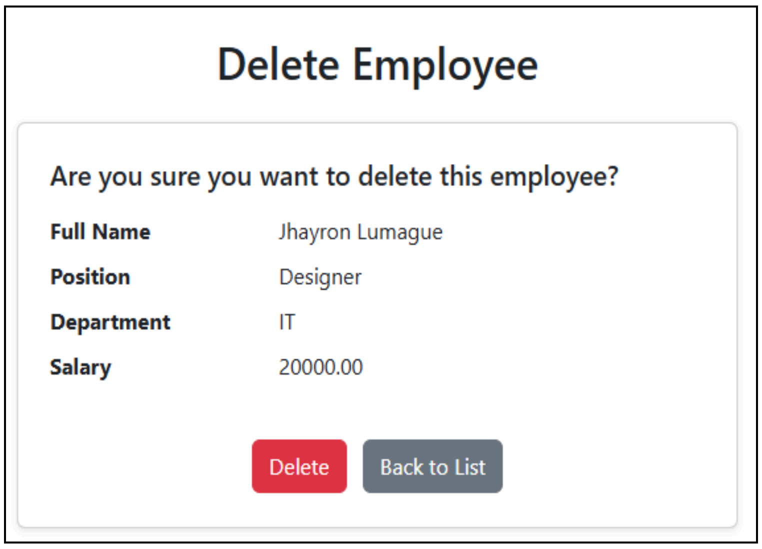
Employee Details

Employee Information

Full Name	Efren Dave Cahilig
Position	Developer
Department	IT
Salary	40000.00

Edit Back to List

DELETE:



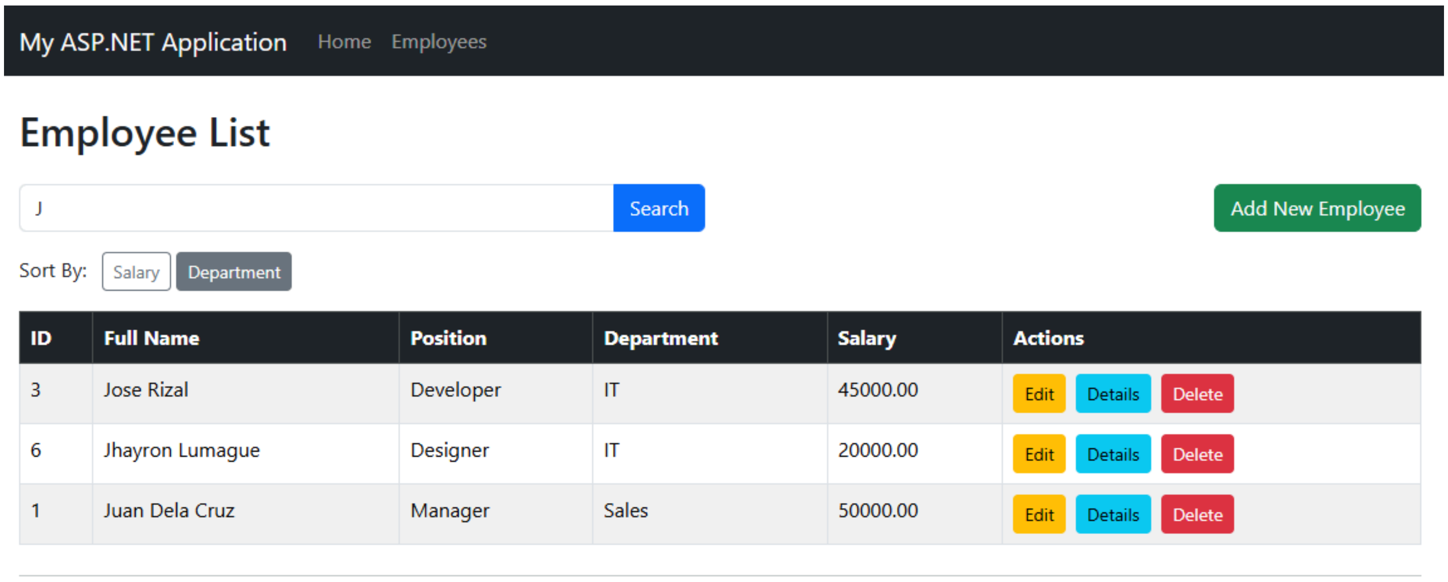
Delete Employee

Are you sure you want to delete this employee?

Full Name	Jhayron Lumague
Position	Designer
Department	IT
Salary	20000.00

Delete Back to List

FILTER BY NAME AND DEPARTMENT



Employee List

J | Search | Add New Employee

Sort By: Salary | Department

ID	Full Name	Position	Department	Salary	Actions
3	Jose Rizal	Developer	IT	45000.00	Edit Details Delete
6	Jhayron Lumague	Designer	IT	20000.00	Edit Details Delete
1	Juan Dela Cruz	Manager	Sales	50000.00	Edit Details Delete

Rubrics

Criteria	Excellent (5)	Good (3)	Poor (1)
Database Creation	Complete with accurate data	Minor issues	Incomplete
MVC Structure	Correct folder organization	Minor mistakes	Disorganized
CRUD Functionality	All working properly	One or two functions not working	CRUD not functional
Entity Framework Integration	Correct model and context setup	Minor configuration error	Not implemented
Documentation/Screenshots	Clear and complete	Missing one part	Incomplete