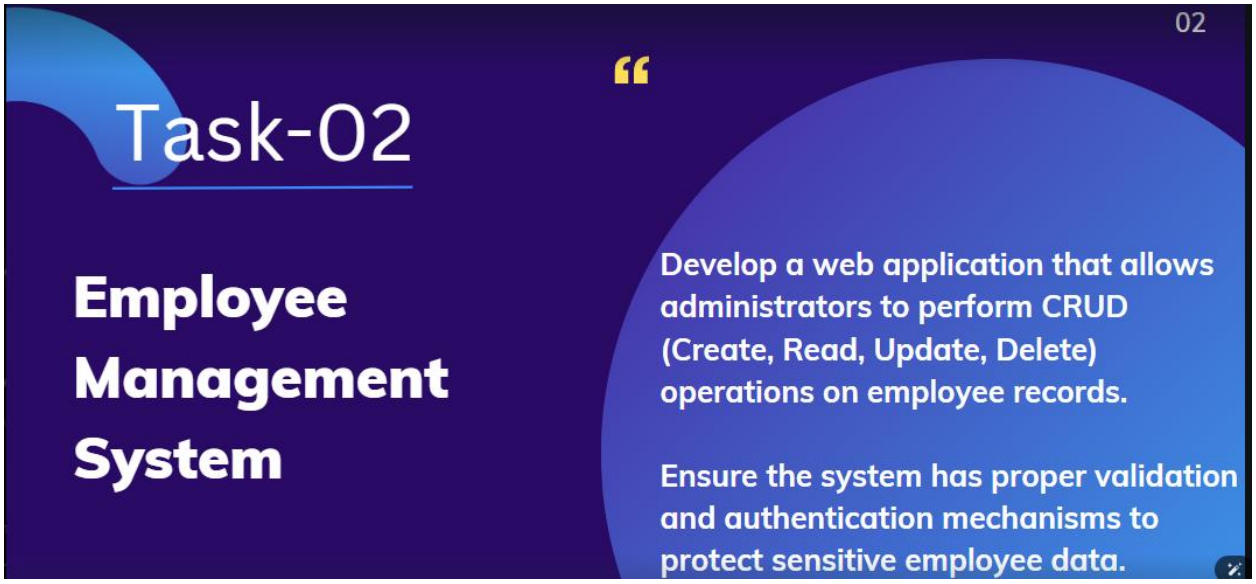
VU22CSEN0400135
N.ALEKHYA

TASK-2



To develop a **web application for an Employee Management System** as outlined in the image, here's a breakdown of what you'll need to do:

Core Requirements

Objective: Create a secure system where administrators can perform CRUD operations on employee records.

Tech Stack Suggestions

You can choose from several modern stacks. Here's one example using **ASP.NET Core** (suitable if you're working with Visual Studio):

- **Backend:** ASP.NET Core (C#)
- **Frontend:** Razor Pages / React / Angular
- **Database:** SQL Server / SQLite

- **Authentication:** ASP.NET Identity
- **ORM:** Entity Framework Core

Features to Implement

1. CRUD Operations for Employees

- **Create:** Form to add a new employee (Name, Age, Role, etc.)
- **Read:** Employee list/dashboard view
- **Update:** Edit employee details
- **Delete:** Remove employee record

2. Validation

- Required field checks
- Email and phone format validation
- Salary range or age constraints

3. Authentication and Authorization

- **Login system** using ASP.NET Identity
- Role-based access control (only admins can manage employees)

4. Security

- Password hashing
- Input sanitization
- HTTPS enforcement
- CSRF protection

Folder Structure Example (ASP.NET Core)

```
/EmployeeManagementSystem
│
├── Controllers
│   └── EmployeesController.cs
│
├── Models
│   └── Employee.cs
│
├── Views
│   └── Employees
│       ├── Create.cshtml
│       ├── Edit.cshtml
│       ├── Index.cshtml
│       └── Delete.cshtml
│
├── Data
│   └── ApplicationDbContext.cs
│
├── wwwroot
│   └── css, js, etc.
```

Step-by-Step: Create Web App in ASP.NET Core

Tools Required:

- **Visual Studio 2022 or later**
- **.NET 6 or .NET 7 SDK**
- **SQL Server** or use **SQLite** for simplicity

Step 1: Create the Project

1. Open **Visual Studio → Create a new project**
2. Choose: **ASP.NET Core Web App (Model-View-Controller)**
3. Name: EmployeeManagementSystem
4. Select:
    - .NET 6 or 7
    - Authentication: **Individual Accounts**
    - Enable Razor runtime compilation (optional)

Step 2: Add the Employee Model

Create a model class in /Models/Employee.cs:

```
using System.ComponentModel.DataAnnotations;

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

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

    [Range(18, 65)]
    public int Age { get; set; }

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

    [EmailAddress]
    public string Email { get; set; }

    [Phone]
    public string Phone { get; set; }

    [Range(10000, 100000)]
    public decimal Salary { get; set; }
}
```

Step 3: Setup EF Core Database Context

In /Data/ApplicationDbContext.cs:

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using EmployeeManagementSystem.Models;

public class ApplicationDbContext : IdentityDbContext
{
   public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
       : base(options)
   {
   }

   public DbSet<Employee> Employees { get; set; }
}
```

Register in Program.cs:

```
builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnec
tion")));
```

Step 4: Scaffold the Employees Controller

Right-click **Controllers → Add → New Scaffolded Item**

- Select: **MVC Controller with views, using Entity Framework**
- Model: Employee
- DbContext: ApplicationDbContext

This auto-generates:

- EmployeesController.cs
- Views: Create, Edit, Delete, Details, Index

Step 5: Secure Access (Admins Only)

Add [Authorize] to the controller:

```
[Authorize(Roles = "Admin")]
public class EmployeesController : Controller
{
    // ...
}
```

You can seed an admin user in Program.cs or via Identity scaffolding.

Step 6: Apply Migrations

Open Package Manager Console:

```
Add-Migration InitialCreate
Update-Database
```

Optional UI Improvements

- Customize the layout in /Views/Shared/_Layout.cshtml
- Add Bootstrap for styling
- Use role-based display in navigation

VU22CSEN0400135
N.ALEKHYA

OUTPUT: