🧁 Part 1 – Cookie-Based Login Page (Concept + Example)

💡الفكرة:

الـ **Cookie-based authentication** معناها إن بعد ما المستخدم يعمل login بنخزن **token** أو **session id** داخل **Cookie** ، والـ browser بيبعتها مع كل request بعد كده علشان السيرفر يعرف إن المستخدم ده مسجل دخول.

🧩 مثال مبسط (ASP.NET Core MVC):

```
// LoginController.cs
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using System.Security.Claims;
public class LoginController : Controller
{
  [HttpGet]
  public IActionResult Index() => View();
  [HttpPost]
  public async Task<IActionResult> Index(string username, string password)
  {
    // مجرد مثال - المفروض تتحقق من الداتا بيز
    if (username == "admin" && password == "123")
    {
      var claims = new List<Claim>
      {
        new Claim(ClaimTypes.Name, username)
      };
      var identity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);
      var principal = new ClaimsPrincipal(identity);

      await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, principal);
      return RedirectToAction("Dashboard", "Home");     }
```

```
    ViewBag.Error = "Invalid login credentials.";
    return View();
  }
  [HttpGet]
  public async Task<IActionResult> Logout()
  {
    await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
    return RedirectToAction("Index");
  }
}
// Startup.cs or Program.cs
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
  .AddCookie(options =>
  {
    options.LoginPath = "/Login/Index";
    options.ExpireTimeSpan = TimeSpan.FromMinutes(30);
  });
```

النتيجة:

- المستخدم بيسجل الدخول.
- السيرفر بيعمل Cookie فيها معلومات الهوية.
- في كل request بعد كده، السيرفر بيتعرف عليه تلقائيًا.

Part 2 – 3 Target Software Houses (and Why)

1 Link Development

- واحدة من أكبر الشركات في مصر والشرق الأوسط في مجال التحول الرقمي.
- بتشتغل بتقنيات Microsoft Stack (ASP.NET, Azure, Dynamics).
- بيهتموا بالـ Enterprise solutions وبيطبقوا هندسة برمجية قوية (Repository, Service Layer, Unit of Work).

2 ITWorx

- شركة مشهورة بمشاريع تعليمية وحكومية كبيرة.
- عندهم structure نظيف جدًا وبيستخدموا Architecture Layers واضحة.
- مناسب جدًا لو عايز تطور مهاراتك في تحليل الأنظمة الضخمة.

3 Vezeeta

- شركة برمجيات طبية.(HealthTech)
- بتركز على الأداء العالي، scalability، والتكامل بين الأنظمة.
- هتتعلم فيها best practices فيREST APIs ، microservices، و security.

⚙️ Part 3 – Report about Service Layer

📘 1. Concept

الـ **Service Layer**هي طبقة من الـ architecture بتفصل الـ **business logic** عن باقي أجزاء النظام.
بتكون حلقة وصل بين:

- الـController (Presentation Layer)
- والـRepository Layer (Data Access Layer)

📊 2. Purpose

- تخلي الكود أنضف وأوضح.(Separation of Concerns)
- تسمح بإعادة استخدام الكود بسهولة.
- تسهل عملية الاختبار.(Unit Testing)
- تمنع الـ Controller من التعامل المباشر مع الداتا بيز.

🧠 3. Structure

Controllers → Services → Repositories → Database

🛠️ 4. Example Implementation

```
// IRepository.cs
public interface IUserRepository
{
  User GetByUsername(string username);
}
```

```
// UserRepository.cs
public class UserRepository : IUserRepository
{
  private readonly AppDbContext _context;
  public UserRepository(AppDbContext context) => _context = context;
  public User GetByUsername(string username)
    => _context.Users.FirstOrDefault(u => u.Username == username);
}
// IUserService.cs
public interface IUserService
{
  bool ValidateUser(string username, string password);
}
// UserService.cs
public class UserService : IUserService
{
  private readonly IUserRepository _repo;
  public UserService(IUserRepository repo) => _repo = repo;
  public bool ValidateUser(string username, string password)
  {
    var user = _repo.GetByUsername(username);
    return user != null && user.Password == password;
  }}
```

💬 5. Summary

- **Repository Layer** → مسؤولة عن التعامل مع الـDatabase.
- **Service Layer** → فيها منطق العمل.(Business Logic)
- **Controller** → فقط يستقبل الطلب ويستدعي الـService.

ده بيخلي المشروع modular ، قابل للتوسعة، وسهل الصيانة.