1️⃣ كيف نتعامل مع مشكلة أمان Connection String المكتوبة كـHardcode ؟

لما نكتب connection string جوّه الكود، ده خطر جدًا لأن أي حد ممكن يشوفها.
عشان نحميها، نستخدم واحدة من الطرق دي👇

3 ✅طرق لحماية:Connection String

1. **Configuration File**
   نحطها في ملف appsettings.jsonبدل الكود، ونقرأها من هناك.

```
"ConnectionStrings": {
"DefaultConnection": "Server=.;Database=MyDB;Trusted_Connection=True;"
}
```

ونستدعيها في الكود:

```
var conn = builder.Configuration.GetConnectionString("DefaultConnection");
```

2. **Environment Variables**
   نخزنها في متغير بيئة (Environment Variable) على السيرفر أو الجهاز.

كده محدش يقدر يشوفها في الكود أو ملفات المشروع.

3. **User Secrets / Azure Key Vault**
   نستخدم أدوات تخزين سرّية زي:

) **User Secrets**في الـ(development

) **Azure Key Vault**في الإنتاج(

ودي بتشفر القيم وبتخلي الوصول ليها آمن.

2️⃣ ما هوAJAX ؟

AJAX = **Asynchronous JavaScript And XML**
ببساطة:
هو طريقة لتحديث جزء من الصفحة **بدون إعادة تحميل الصفحة بالكامل**.

3 ✳️أسباب لاستخدام AJAX في:MVC

1. يحسّن **سرعة الأداء** لأن الصفحة مش بتتعمل.Reload
2. يحسّن **تجربة المستخدم.(UX)**
3. يقلّل **استهلاك السيرفر** لأن الطلبات أخف وأصغر.

⚙️ Demo

بسيط في MVC:

```
<!-- View -->
<button id="loadBtn" class="btn btn-primary">Load Students</button>
<div id="result"></div>

@section Scripts {
<script>
$("#loadBtn").click(function () {
$.ajax({
url: "/Student/GetStudents",
type: "GET",
success: function (data) {
$("#result").html(data);
} });   });
</script>}
// Controller
public IActionResult GetStudents()
{
var students = _studentRepo.GetAll().Select(s => s.Name).ToList();
return PartialView("_StudentsList", students);
}
<!-- Partial View (_StudentsList.cshtml) -->
<ul>
@foreach (var name in Model)
{
<li>@name</li>
}
</ul>
```

🧱 3 (Bonus) LinkedIn Post — Creational Design Patterns

✍️بالعربي بأسلوب بسيط

#DesignPatterns 🚀بالعربي — الجزء الأولCreational Patterns :

لو بدأت تتعمق في البرمجة الكائنية، أكيد سمعت عن *Design Patterns*.
النهارده هنتكلم عن نوع مهم جدًا منهم:
— **Creational Design Patterns** 💡أنماط الإنشاء

🎯الهدف منها:
تنظيم وإنشاء الكائنات (Objects) بطريقة ذكية بدل ما نستخدم new في كل مكان.
ليه؟
علشان الكود يكون **قابل للتوسع، قابل للاختبار، وسهل الصيانة**.

🧩أشهر الأنواع:

1. **Singleton**
   يضمن إن الكلاس بيتعمل منه كائن واحد فقط) زي DB Context أو Logger).
2. **Factory Method**
   ينشئ الكائن المناسب حسب الحالة، بدل ما نكتب new لكل نوع.
3. **Dependency Injection (DI)**
   بدل ما الكلاس ينشئ Dependencies بنفسه، السيستم بيحقنها له ASP.NET Core) بيستخدمها بشكل أساسي.)

🔥**ليه نستخدمها؟**

- بتقلل الـ Coupling بين الكلاسات.
- بتخلي الكود مرن جدًا وسهل التعديل.
- بتخلي الـ Testing أسهل.