امیرحسین دوستی

**1. تعریف UML**

UML (مخفف **Unified Modeling Language**) یعنی زبان مدل‌سازی یکپارچه. یه زبان تصویریه برای طراحی و مستندسازی سیستم‌های نرم‌افزاری. باهاش می‌تونیم اجزای مختلف سیستم مثل کلاس‌ها، ارتباط‌ها، رفتارها و ... رو به صورت نمودار نشون بدیم.

**2. فهرست نمودارهای UML**

UML در کل 14 تا نمودار داره که به دو دسته تقسیم می‌شن:

* **نمودارهای ساختاری (ایستا):**
  1. Class Diagram
  2. Object Diagram
  3. Component Diagram
  4. Deployment Diagram
  5. Package Diagram
  6. Composite Structure Diagram
  7. Profile Diagram
* **نمودارهای رفتاری (پویا):**
  1. Use Case Diagram
  2. Activity Diagram////////////////////////////////////
  3. State Machine Diagram (یا State Diagram)
  4. Sequence Diagram
  5. Communication Diagram
  6. Timing Diagram
  7. Interaction Overview Diagram

**3. کاربرد UML**

* طراحی اولیه‌ی سیستم‌ها
* مستندسازی پروژه‌ها
* درک بهتر از نیازها و رفتار سیستم
* ارتباط بین اعضای تیم برنامه‌نویسی
* کمک به طراحی دیتابیس و معماری نرم‌افزار

**4. مزایا**

* استاندارد جهانیه
* گرافیکیه و فهمش راحته
* کمک می‌کنه قبل از کدنویسی ساختار سیستم رو بفهمیم
* واسه همه نوع پروژه کاربرد داره (چه وب، چه دسکتاپ)
* باعث کاهش خطا در طراحی می‌شه

**5. معایب**

* ممکنه پیچیده بشه اگه زیاد جزئیات بدیم
* برای پروژه‌های خیلی کوچیک ممکنه استفاده از UML وقت‌گیر باشه
* گاهی ابزارهای طراحی UML سنگین یا سخت‌فهم هستن

**6. جنبه‌های ایستا و پویا**

* **ایستا (Static):** بخش‌هایی از سیستم که تغییر نمی‌کنن، مثلاً کلاس‌ها، ارتباط‌ها، ساختارها (مثلاً Class Diagram).
* **پویا (Dynamic):** رفتار سیستم در طول زمان و شرایط مختلف، مثلاً گردش کار، واکنش به رویدادها (مثل Sequence Diagram).

**7. نمودارهای زبان UML (تعریف + مثال)**

**1. Use Case Diagram**

**تعریف:** تعامل بین کاربران (اکتور) و سیستم رو نشون می‌ده.  
**مثال:** سیستم کتابخانه – کاربر می‌تونه "کتاب جستجو کنه"، "کتاب قرض بگیره".  
**توضیح:** این نمودار نشون می‌ده هر کاربر چه کاری تو سیستم می‌تونه انجام بده.

**2. Activity Diagram**

**تعریف:** گردش کار یا فرایند رو مرحله به مرحله نشون می‌ده.  
**مثال:** ثبت‌نام در سایت – ورود اطلاعات → تایید ایمیل → ثبت موفق  
**توضیح:** مراحل مختلف یک فرآیند رو با جهت فلش نشون می‌دیم.

**3. State Diagram**

**تعریف:** تغییر وضعیت یک شی در طول زمان و بر اساس رویدادها رو نشون می‌ده.  
**مثال:** وضعیت سفارش – جدید → پردازش → ارسال → تحویل  
**توضیح:** هر وضعیت و رویدادی که باعث تغییر اون بشه، نمایش داده می‌شه.

**4. Communication Diagram**

**تعریف:** تعامل بین اشیاء و ترتیب پیام‌ها رو نشون می‌ده.  
**مثال:** خرید از سایت – کاربر پیام "درخواست سفارش" به سیستم می‌فرسته، سیستم با انبار چک می‌کنه.  
**توضیح:** اشیاء با خط به هم وصلن و ترتیب پیام‌ها با شماره مشخص می‌شه.

**5. Sequence Diagram**

**تعریف:** ترتیب اجرای پیام‌ها بین اشیاء در طول زمان رو نشون می‌ده.  
**مثال:** لاگین – کاربر نام کاربری و رمز وارد می‌کنه، سیستم بررسی می‌کنه، اگه درست بود پیام موفق برمی‌گردونه.  
**توضیح:** پیام‌ها از بالا به پایین و به ترتیب زمانی هستن.

**6. Timing Diagram**

**تعریف:** تغییرات وضعیت یک یا چند شی رو بر حسب زمان نشون می‌ده.  
**مثال:** وضعیت یک سنسور دما در طول زمان – روشن، گرم، خاموش  
**توضیح:** محور افقی زمانه، محور عمودی وضعیت‌ها.

**7. Interaction Overview Diagram**

**تعریف:** ترکیب نمودار فعالیت و ترتیب برای نشون دادن تعاملات کلی  
**مثال:** فرایند پرداخت – شامل لاگین، انتخاب کالا، پرداخت  
**توضیح:** هر بخش می‌تونه یه نمودار Sequence یا Activity باشه.

**8. Profile Diagram**

**تعریف:** سفارشی‌سازی UML با ایجاد stereotypeها و ویژگی‌های خاص  
**مثال:** تعریف یک نوع کلاس خاص به اسم «سیستم حساس به زمان»  
**توضیح:** بیشتر برای توسعه‌دهنده‌های ابزارهای UML کاربرد داره.

**9. Composite Structure Diagram**

**تعریف:** ساختار داخلی کلاس‌ها و ارتباط اجزای داخلی  
**مثال:** کلاس ماشین که شامل موتور، چرخ، ترمز هست  
**توضیح:** نشون می‌ده یک کلاس از چه قسمت‌هایی تشکیل شده.

**10. Class Diagram**

**تعریف:** کلاس‌ها، ویژگی‌ها، متدها و روابط بین اونا  
**مثال:** کلاس "دانشجو" با ویژگی‌هایی مثل نام، شماره دانشجویی  
**توضیح:** این نمودار پایه‌ای‌ترین و معروف‌ترین نمودار UML هست.

**11. Package Diagram**

**تعریف:** گروه‌بندی کلاس‌ها در پکیج‌ها  
**مثال:** پکیج "کاربران"، پکیج "محصولات"، پکیج "پرداخت‌ها"  
**توضیح:** ساختار کلی و گروه‌بندی کلاس‌ها رو نشون می‌ده.

**12. Object Diagram**

**تعریف:** نمونه‌هایی از کلاس‌ها (اشیاء) و وضعیت فعلی اونا  
**مثال:** شیء علی از کلاس دانشجو با مقدار "علی" برای نام  
**توضیح:** یه اسنپ‌شات از لحظه‌ای خاص در سیستم

**13. Component Diagram**

**تعریف:** اجزای نرم‌افزار و ارتباط بینشون  
**مثال:** کامپوننت "واسط کاربر"، "منطق کسب‌وکار"، "دیتابیس"  
**توضیح:** در معماری نرم‌افزار استفاده می‌شه.

**14. Deployment Diagram**

**تعریف:** چطوری اجزای نرم‌افزار روی سخت‌افزارها قرار می‌گیرن  
**مثال:** سرور وب، سرور دیتابیس، کامپیوتر کاربر  
**توضیح:** نحوه استقرار سیستم رو نشون می‌ده.

**1. ASP**

مثل یه روش قدیمی برای ساختن صفحات وبه. یه جوریه که وقتی کاربر یه صفحه رو باز می‌کنه، سرور میاد یه صفحه مخصوص همون لحظه براش می‌سازه. مثلاً می‌تونه اطلاعات از دیتابیس بخونه و توی صفحه نشون بده.

**2. .NET**

یه جعبه ابزار بزرگه که مایکروسافت ساخته. توش کلی ابزار هست که باهاش می‌تونیم برنامه‌های مختلفی درست کنیم: برنامه دسکتاپ، موبایل، وب و غیره. مثل یه عالمه وسیله توی یه کیف ابزار برای برنامه‌نویسا!

**3. ASP.NET Core**

نسخه جدید و باحال‌تر ASP هست. سبکه، سریع‌تره و می‌تونه روی ویندوز، لینوکس یا مک اجرا بشه. باهاش میشه وب‌سایت، وب‌اپ و API ساخت. خلاصه یه جور جدید و حرفه‌ای‌تر برای ساختن سایت‌هاست.

**4. Middleware**

اینا مثل نگهبان‌هایی هستن که توی مسیر درخواست کاربر تا رسیدن به جواب وایسادن. مثلاً یکی‌شون چک می‌کنه که کاربر لاگین کرده یا نه، یکی دیگه لاگ می‌گیره، یکی دیگه اگه خطا بود، نشونش می‌ده. هر Middleware یه کاری می‌کنه.

**5. Web Routing**

وقتی تو آدرس مرورگر یه چیزی می‌نویسی (مثلاً /products/2) این سیستم میاد می‌فهمه باید کدوم قسمت از برنامه اجرا بشه. یه جور راه‌یابی برای درخواست‌هاست.

**6. Data Annotation**

یه‌سری برچسب هستن که روی مدل‌ها (کلاس‌ها) می‌زنیم تا یه چیزهایی مشخص کنیم. مثلاً بگیم این فیلد نباید خالی باشه، یا حداکثر تعداد کاراکتر چنده. برای کنترل اعتبار اطلاعات خیلی کاربردیه.

**7. DbContext**

یه چیزی مثل پل ارتباطی بین برنامه و دیتابیسه. ما با می‌تونیم بگیم چه اطلاعاتی رو بخون، ذخیره کن، حذف کن یا تغییر بده. یه جور مغز عملیات دیتابیسیه.

**8. Connection String**

مثل یه آدرس یا رمز ورود به دیتابیسه. داخلش مشخص می‌کنیم به کدوم دیتابیس وصل بشیم، روی کدوم سرور، با چه یوزرنیم و رمزی. برنامه با این آدرس می‌فهمه چطور وصل شه به دیتابیس.

**9. Dependency Injection**

یه روش خیلی باحاله که توش ما به جای اینکه خودمون کلاس‌ها رو بسازیم، برنامه خودش اونا رو برامون می‌سازه و می‌ده دستمون. اینطوری هم کدمون مرتب‌تر میشه هم قابل تست‌تر.

**10. Controller**

یه قسمتی از برنامه‌ست که وقتی کاربر یه درخواستی می‌فرسته، میاد اون درخواست رو بررسی می‌کنه، اطلاعات لازم رو می‌گیره و جواب مناسب رو برمی‌گردونه. انگار مغز برنامه‌ست که تصمیم می‌گیره چی بشه.

**11. ORM**

مخفف **Object Relational Mapping** هست. یه چیزی که کمک می‌کنه با دیتابیس راحت‌تر کار کنیم. به جای اینکه SQL بنویسیم، با کلاس و شیء کار می‌کنیم، خودش می‌فهمه باید چی تو دیتابیس انجام بده.

**12. Entity Framework**

یکی از معروف‌ترین ORMها برای زبان C# هست. با EF می‌تونیم خیلی راحت اطلاعات رو از دیتابیس بخونیم، ذخیره کنیم و غیره. بدون اینکه خودمون مستقیم SQL بنویسیم.

**13. Razor**

یه زبان ساده و قاطی‌شده از HTML و C# هست که توی Viewها استفاده می‌کنیم. یعنی هم می‌تونیم کد HTML بنویسیم، هم کد C# وسطش بچسبونیم. خیلی راحت برای ساخت صفحات داینامیکه.

**14. Validation**

یعنی بررسی کنیم که اطلاعاتی که کاربر وارد کرده درسته یا نه. مثلاً مطمئن بشیم که اسم خالی نیست، یا سن عدد مثبته. این کار رو می‌تونیم با Data Annotation هم انجام بدیم.