1. Domain driven design چیست؟
   1. تعریف ساده اش اینه: شیوه ای برای توسعه سیستم های پیچیده. منظور اینه که ما یه سری مسئله سمت طراحی و ایجاد سیستم مورد نظرمون داریم که میخوایم اینا رو برطرف کنیم. DDD میاد و یه سری مفاهیم، راهنما و دستورالعمل معرفی می کنه که بر اساس اونا code base مونو ایجاد کنیم.
2. منظور از domain اصلا چیه؟
   1. کلیتش اینه: یه موضوعی وجود داره که حول محور اون میخوایم یه برنامه بنویسیم. Domain میاد و میگه که اطلاعات و دانش مرتبط با اون موضوع و منطق برنامه (application logic) به چه صورته؛ اینکه چه چیز هایی داریم، چه اتفاقاتی میفته، چه اتفاقاتی ممکنه بیفته، در مواجهه با هر اتفاق چه تصمیمی اتخاذ کنیم و...
   2. یه مثال میزنیم: فرض کنیم یه برنامه meet داریم. یه سری افراد دارن داخل این برنامه یه جلسه یا میتینگ برگزار می کنن. هر کس که حضور داره میتونه یه وضعیتی داشته باشه مثل online, disconnect, early left, …. توی هر کدوم از این وضعیت ها ما میتونیم دقیق مشخص کنیم که اگه شخص قرار بگیره، چه اتفاقاتی باید بیفته. حالا مثلا early left رو در نظر بگیریم؛ طرف تا چه مدت زمانی از میتینگ بیرون بره، ما توی این وضعیت نگه اش می داریم؟ یه دیقه، دو دیقه، ده دیقه یا هر مقدار دیگه ای. این ویژگی و تصمیمی که براش می گیریم، همه در حیطه domain هستن.
3. مزایای استفاده از DDD چیه؟
   1. یه متدولوژی ثابت شده اس در طراحی سیستم های پیچیده به خاطر دستور العمل ها و راهنمایی های گام به گام اش در طراحی و پیاده سازی سیستم ها.
   2. در طولانی مدت، سیستم ما انعطاف پذیری بیشتری داره. راحتتر میتونیم تغییرات رو بهش اضافه یا اعمال کنیم.
   3. از معماری هایی مثل میکروسرویس یا سیستم های توزیع شده پشتیبانی می کنه (microservice and distributed systems)
   4. تست نویسی به مراتب راحتتره (نداشتن وابستگی های infrastructure و مشخص بودن مرز بین قسمت های مختلف)
   5. چون قسمت های مختلف domain رو جدا و هر بخش رو مجزا هندل می کنه، نگهداری (maintenance) پروژه راحتتره.
   6. گرچه در ابتدا به خاطر پیچیدگی و نداشتن پایه اولیه پروژه، پیاده سازی و توسعه طولانی تر و سختتره اما در طولانی مدت ساده تر و کم هزینه تر میشه.
4. 2 بخش اصلی DDD چی هستن؟
   1. Strategic desgin: این بخش گرچه به نسبت بخش بعدی کمتر به چشم میاد، اما به شدت مهمه. تو این بخش ما domain رو بررسی می کنیم و سعی میکنیم بفهمیم subDomain های ما چیا هستن، چه ارتباطی با هم دیگه دارن، چرا اصلا وجود دارن (در واقع میخوایم بدونیم هر subDomain چه نقش کلیدی ای رو داره بازی میکنه که بودنش رو واجب میکنه)، هر subDomain رو کجا قرار میدیم (ممکنه چندتاشون با هم داخل یه برنامه باشن و یک یا چندتا داخل یه برنامه دیگه)، میزان اهمیت و هزینه توسعه هر subDomain و... . به طور کلی، میپرسیم که چه چیزهایی داخل domain ما هستن و چرایی وجودشون؛ اینکه چه مشکلی رو دارن برای ما حل می کنن که به خاطرش بخوایم اینا رو داشته باشیم. در نهایت، هر subDomain باید به یه logical self-contained component تبدیل بشه که در درون خودش، داره مسائل مربوط به اون subDomain رو حل می کنه و با subDomain های دیگه هم در ارتباطه.
   2. Tactical design: اینجا حالا با "چطور" سر و کار داریم. چطور یه subDomain کار میکنه؟ چطور با subDomain ها ارتباط برقرار میکنه؟ چطور میخواد مسائل رو حل کنه؟ جواب این سوال ها رو توی این بخش میدیم که بیشتر با مفاهیم کدنویسی و فریمورک هاش سروکار داره.