الف) مسأله:

یک سامانه مدیریت درمانگاه که امکان تخصیص وقت ملاقات به بیماران را فراهم میکند. در این سامانه :

۱. هر پزشک (که یا عمومی است یا متخصص) می‌تواند چندین قرار ملاقات (ویزیت) داشته باشد.
۲. زمان های مجاز قرار ملاقات، شنبه تا چهارشنبه ۹ الی ۱۸ است.
۳. زمان هر قرار ملاقات پزشک عمومی بین ۵ تا ۱۵ دقیقه و پزشک متخصص بین ۱۰ الی ۳۰ دقیقه است.
٤. هر پزشک طبق یک برنامه هفتگی در درمانگاه حضور دارد و قرار ملاقات های تنظیم شده با هر پزشک، باید در بازه زمانی حضور پزشک در درمانگاه باشد.
٥. قرار ملاقات های یک بیمار نباید با هم همپوشانی داشته باشد.
٦. یک بیمار در یک روز، بیش از ۲ قرار ملاقات نداشته باشد.
۷. حداکثر ۲ قرار ملاقات یک پزشک عمومی و ۳ قرار ملاقات یک پزشک متخصص میتواند با هم همپوشانی داشته باشد.

ب) توسعه های آتی :

پیاده سازی را با لحاظ کردن این موارد در توسعه های آتی انجام دهید (در حال حاضر نیاز نیست):

۱. در مورد قرار ملاقات خصوصیات دیگر (مثل اتاق ویزیت) به سامانه اضافه شود و در تنظیم قرارها لحاظ شود.
۲. نوع پزشک دیگر (مانند "فوق تخصص") هم به سیستم اضافه شود.
۳. خبردهی قرار ملاقات به پزشک و بیمار به صورت ایمیل یا پیامک
٤. اطلاع رسانی به پزشک و بیمار در صورتی که یک قرار ملاقات تغییر کرد
٥. نگهداری اطلاعات فردی و رزومه پزشک، سابقه ویزیت‌های یک پزشک، نظر سنجی از بیماران ملاقات شده درباره پزشک معالج
٦. نگهداری مشخصات فردی بیمار، سوابق ویزیت های بیمار، پرونده پزشکی بیمار
۷. محاسبات مربوط به صورتحساب بیمار و حق الزحمه پزشک و تعاملات با شرکت های بیمه

ج) خروجی مورد انتظار:

۱. سرویس ثبت یک قرار ملاقات بوسیله کاربر در یک زمان مشخص :

```
SetAppointment (Doctor, Patient, DurationMinutes, StartDateTime)
```

۲. سرویس ثبت خودکار یک قرار ملاقات در اولین بازه زمانی ممکن برای یک پزشک و یک بیمار:

```
SetEarliestAppointment (Doctor, Patient, DurationMinutes)
```

۳. پیاده سازی Unit Test (برای دامین)
٤. بکارگیری رویکرد DDD در پیاده سازی
٥. پیاده سازی integration test برای سرویس‌ها
٦. پیاده سازی تست های acceptance مطابق رویکرد BDD

* کد را بر روی GitHub تحویل دهید به گونه ای که منعکس کننده سابقه توسعه و تغییرات انجام شده (Commit History) نیز باشد.
* ترجیح این است که پاسخ شما به تمرین کامل باشد، چنانچه – به دلیل کمبود وقت یا دلایل دیگر – هرکدام از موارد الف، ب و ج را پیاده سازی/لحاظ نکردید، در پاسخ ارسالی (ReadMe) مشخص کنید. در صفحه بعد موارد مورد توجه در ارزیابی بیان شده‌اند.

در ارزیابی پاسخ شما به این موارد (اما نه صرفا محدود به آنها) توجه می‌شود :

صحت و درستی پیاده سازی: بررسی می‌شود که آیا کد عملکرد توصیف شده مورد نیاز را به درستی اجرا می کند یا خیر؛ برنامه ریزی قرار ملاقات، در دسترس بودن پزشک و سایر قوانین ذکر شده به درستی پیاده سازی شده‌اند؟

خوانایی و قابلیت نگهداری (Clean Code): خوانایی و قابلیت نگهداری کد، وضوح، نامگذاری مناسب، قالب بندی ثابت و استفاده بجا از Comment برای توضیح منطق پیچیده یا نکات مهم تصمیم گیری.

Test: آیا unit test های مناسب برای ارزیابی صحت عملکرد و رفتار سیستم پیاده سازی شده است ؟ (توجه به جزییات و در نظر گرفتن انواع حالت‌های مختلف برای سنجش میزان دقت و دید جامع شما برای ما بسیار مهم است). به موارد حدی (نزدیک محدوده) توجه شده است؟ به همین ترتیب جامعیت و صحت تست‌های Integration و Acceptance مورد توجه قرار می‌گیرند.

Separation Of Concerns و SOLID: آیا کد از اصول Separation of Concerns و SOLID پیروی می کند یا خیر؟ آیا مسئولیت ها به طور مناسب بین کلاس ها و متد ها تقسیم شده اند و آیا encapsulation به درستی رعایت شده است یا خیر؟

الگوهای طراحی: آیا الگوهای طراحی بجا و به درستی در کد استفاده شده اند ؟ آیا ساختار کد به اصول و مزایای هر الگوی طراحی پایبند است؟

Error Handling: کد چگونه خطاها و exception ها را مدیریت می کند؟ مکانیسم‌های مناسب exception handling وجود دارد و پیام‌های خطا مناسب و کاربردی هستند؟

Commit History : توجه به commit message ها و تناسب آنها، تغییرات کد در هر مرحله که منعکس کننده رویکرد حل مسأله و رفع bug ها و توالی commit ها که منعکس کننده روند حل مسأله است.

توسعه پذیری: آیا کد امکان گسترش یا اصلاح آسان در آینده را می دهد. آیا اضافه کردن انواع جدید موجودیت ها یا اضافه کردن عملکردهای اضافی (به خصوص موارد ذکر شده در توسعه ها آتی) بدون تغییرات قابل توجه در کد موجود انجام می‌شود؟

توجه به Best Practice : آیا کد از بهترین شیوه ها و تجربیات خاص برای Stack انتخاب شده (C#, ASP.NET) پیروی می کند؟ مواردی مانند استفاده صحیح و مناسب از Data structure ها، استفاده درست از interface ها، استفاده مناسب از EF Core، async/wait و ...

رعایت اصول DDD:

- شناسایی صحیح Business Context ها، مرز بندی مناسب و جلوگیری از نشت منطق از یک BC به سایرین
- ایجاد و بکارگیری زبان مشترک (Ubiquitous Language) بین دو حوزه مسأله و راه حل
- استفاده مناسب از تکنیک‌های context mapping برای ایجاد روابط و مرزهای بین BC های مختلف؛ استفاده مناسب از الگوهای مرتبط برای تعریف همکاری‌ها و وابستگی ها
- بکارگیری مناسب aggregate boundary برای encapsulate کردن موجودیت‌های مرتبط و الزام به consistency در داخل مرزهای هریک و جلوگیری از دسترسی غیرمجاز به داخل aggregate ها
- شناسایی و طبقه بندی صحیح انواع موجودیت ها (aggregates, root aggregates, value object)
- شناسایی درست domain service ها تمایز آنها از سایر رفتارها
- پیاده سازی درست و اطمینان از برقراری business invariant ها
- توزیع مناسب رفتار به طور مناسب بین انواع موجودیت ها و اجتناب از anemic domain
- شناسایی و مدیریت صحیح درست domain event ها