1.سناریو: مدیریت سیستم رزرو تاکسی

•شما باید سیستمی طراحی کنید که کاربران بتوانند تاکسی رزرو کنند، رانندگان بتوانند وضعیت تاکسی خود را به‌روزرسانی کنند، و مدیریت کلی سیستم به‌صورت شی‌گرا و مطابق با اصول SOLID باشد.

.

1.نیازمندی‌ها

1.1. Single Responsibility Principle (SRP):

•هر کلاس باید فقط یک مسئولیت داشته باشد.

1.یک کلاس برای کاربر (User) با مسئولیت مدیریت اطلاعات کاربر.
2.یک کلاس برای تاکسی (Taxi) با مسئولیت مدیریت وضعیت تاکسی‌ها.
3.یک کلاس برای رزرو (Booking) با مسئولیت مدیریت فرایند رزرو.

2.1. Open/Closed Principle (OCP):

•کلاس‌ها باید برای توسعه باز و برای تغییر بسته باشند.

1.باید بتوانید به‌راحتی انواع جدیدی از رزرو یا تاکسی را اضافه کنید بدون تغییر در کد موجود.

3.1. Liskov Substitution Principle (LSP):

•کلاس‌های فرزند باید بتوانند جایگزین کلاس والد شوند.

1.اگر کلاس Taxi یک کلاس پایه برای انواع خاصی از تاکسی‌ها (مانند TaxiXL یا SharedTaxi) است، این اصل را رعایت کنید.

4.1. Interface Segregation Principle (ISP):

•کلاس‌ها نباید مجبور شوند متدهایی را که استفاده نمی‌کنند پیاده‌سازی کنند.

1.از چندین اینترفیس کوچک به جای یک اینترفیس بزرگ استفاده کنید. به عنوان مثال، اینترفیس‌هایی برای "پرداخت"، "رزرو"، یا "مدیریت راننده".

5.1. Dependency Inversion Principle (DIP):

•ماژول‌های سطح بالا نباید به ماژول‌های سطح پایین وابسته باشند، بلکه هر دو باید به انتزاع وابسته باشند.

1.به جای وابستگی مستقیم کلاس Booking به Taxi یا User، از اینترفیس‌ها و دیزاین پترن‌های مثل Factory یا Dependency Injection استفاده کنید.

تمرین عملی

•کد بالا را پیاده‌سازی کنید.

•از تمام اصول SOLID در طراحی استفاده کنید.
•یک سیستم ساده CLI طراحی کنید که:

•کاربر جدید ایجاد کند.

- تاکسی جدید اضافه کند.
- تاکسی رزرو کند.
- وضعیت تاکسی‌ها را نمایش دهد.
- بررسی کنید که چگونه کلاس‌ها با اصول SOLID گسترش‌پذیری دارند.