Analiza SOLID principa za MediPlan

1. Single Responsibility Principle (SRP)
SRP je zadovoljen. Svaka klasa ima jasno definisanu domenu odgovornosti:

- Korisnik opisuje podatke o korisniku sistema bez obzira na ulogu (doktor, pacijent, administrator).
- Termin se bavi isključivo zakazivanjem i upravljanjem terminima.
- Recenzija opisuje povratne informacije pacijenata o doktorima.
- MedicinskeUsluge modelira medicinske preglede i intervencije.
- HistorijaGenerisanihNalaza vodi evidenciju o poslanim medicinskim nalazima. Zahvaljujući toj podjeli, svaka klasa se može mijenjati nezavisno, isključivo zbog promjene u svojoj odgovornosti.

2. Open-Closed Principle (OCP)
Sistem je dizajniran tako da omogućava proširenje bez izmjene postojećih klasa.
Na primjer:

- Korisnik može biti osnovna klasa, dok se nove uloge (npr. Admin, Pacijent, Doktor) mogu izvesti nasljeđivanjem bez izmjene Korisnik.
- Novi tip medicinske usluge se može dodati kroz novu vrijednost enumeracije VrstaUsluge bez izmjene klase MedicinskeUsluge.
- Ako se dodaju nove funkcionalnosti recenzija (npr. ocjena ponašanja, vrijeme čekanja), moguće je proširiti Recenzija bez narušavanja postojećeg sistema.

3. Liskov Substitution Principle (LSP)
Ako se Korisnik proširi u Pacijent, Doktor ili Admin, svi ti objekti mogu biti korišteni tamo gdje se očekuje Korisnik, npr. u terminima, recenzijama i historiji nalaza, gdje se koristi PacijentId i DoktorId (koji mogu biti povezani s Korisnik).
Time se osigurava poštivanje LSP principa.

4. Interface Segregation Principle (ISP)
Iako trenutno nisu eksplicitno definisani interfejsi u modelu, dizajn omogućava lako razdvajanje funkcionalnosti po tipu korisnika:

- Doktori bi mogli implementirati interfejs kao što je IDoktor, koji sadrži metode za slanje nalaza i pregled zakazanih termina.
- Pacijenti bi mogli koristiti interfejs IPacijent, sa metodama za zakazivanje termina i ostavljanje recenzija.
- Administratori bi imali svoj interfejs IAdministrator, koji sadrži metode za upravljanje korisnicima i uslugama.

Na ovaj način, svaka uloga implementira samo ono što koristi, bez zagušivanja suvišnim funkcijama – što direktno zadovoljava ISP princip i olakšava održavanje sistema.

5. Dependency Inversion Principle (DIP)
Logika zakazivanja termina, slanja notifikacija ili generisanja nalaza može se zasnivati na apstraktnim servisima koje implementiraju konkretne klase. Time Termin ili Recenzija ili ne zavise direktno od npr. e-mail servisa, nego od apstraktnih interfejsa, što omogućava lako testiranje i zamjenu implementacija bez promjene u glavnim klasama.