SOLID PRINCIPI

1. Princip pojedinačne odgovornosti (Single responsibility Principle)

Princip je zadovoljen jer svaka klasa čuva samo njoj bitne podatke, ima samo jednu odgovornost i jedan razlog za promjenu. Nijedna klasa se ne bavi više od jednom svrhom.

2. Open-Closed Principle

Ovaj princip kaže da svaka klasa mora biti otvorena za nadogradnje i zatvorena za modifikacije. Ukoliko promijenimo jednu klasu neće doći do promjene druge klase. Ovaj princip je zadovoljen na našem dijagramu klasa, jer promjena u implementaciji bilo koje klase, ne forsira modifikaciju neke druge klase. Naprimjer, u slučaju da neko ostavi neki lajk ili komentar na neku objavu, korisniku dolazi notifikacija o izvršenoj radnji.

3. Liskov princip zamjene (Liskov Substitution Principle)

Ovaj princip zahtijeva da podklase moraju biti zamjenjive baznoj klasi, tj. da bilo koja upotreba bazne klase omogućava upotrebu i izvedenih klasa, sa istim rezultatom. Naš sistem ne sadrži nasljeđivanja, stoga je ovaj princip zadovoljen.

4. Princip izoliranja interfejsa (Interface Segregation Principle)

Klijenti ne treba da ovise o metodama koje neće upotrebljivati.

Kako smo dizajnirali model sistema, nismo koristili interfejse, te znamo da ovaj princip nije narušen.

5. Princip inverzije ovisnosti (Dependency Inversion Principle)

Ovaj princip zahtjeva da moduli visokog nivoa ne bi trebali ovisiti od modula niskog nivoa. Oba bi trebalo da ovise od apstrakcija.
B. Moduli ne bi trebali ovisiti od detalja. Detalji bi trebali biti ovisni od apstrakcija.

Kako u našem dijagramu nemamo nasljeđivanje ovaj princip je zadovoljen. Također, s obzirom da u aplikaciji nema tolike kompleknosti da bi se nešto moglo nazvati apstrakcijom pa da zavisi od detalje i s obzriom da se ne mogu primjetiti high-level i low-level moduli, ovaj princip je zadovoljen.