S – Single Responsibility Principle

Ovaj princip nalaže da svaka klasa treba imati samo jednu odgovornost, odnosno samo jedan razlog za promjenu. Klasa **Korisnik** poštuje princip jedinstvene odgovornosti jer se bavi isključivo podacima i funkcionalnostima korisnika sistema. Iako sadrži više atributa, svi su vezani za jedinstvenu svrhu tj. reprezentaciju korisnika sistema. Klasa **Dijete** ima jasnu odgovornost, koja je predstavljanje podataka i operacije vezane za dijete u vrtiću. Klasa **Grupa** ima samo naziv i služi za organizaciju djece i aktivnosti, čime ostaje fokusirana na jednu odgovornost. Klasa **Aktivnost** predstavlja jednu aktivnost u vrtiću i uključuje atribute i metode direktno vezane za njeno trajanje, opis i učestvovanje, bez uplitanja u druge funkcionalnosti, čime takođe poštuje princip. Klasa **Obavijest** odgovara samo za kreiranje i prikaz obavijesti, što je jasna i ograničena odgovornost. Klasa **Prisustvo** ima fokus isključivo na evidenciju prisustva, status i razlog odsustva, čime poštuje princip jedinstvene odgovornosti.

O – Open/Closed Principle

Ovaj princip kaže da bi klasa trebala biti otvorena za proširenje, ali zatvorena za izmjene. Iako naš sistem nije implementiran s nasljeđivanjem poštujemo princip tako što su klase modelirane tako da omogućavaju proširenje funkcionalnosti bez potrebe za izmjenom postojećeg koda. Funkcionalnosti se dodaju kroz nove metode bez diranja postojećih klasa koje su već u upotrebi. Ovaj pristup omogućava laku adaptaciju sistema na buduće promjene.

L – Liskov Substitution Principle

Liskov princip nalaže da objekti izvedenih klasa mogu biti zamijenjeni objektima svojih baznih klasa bez promjene u ponašanju sistema. U ovom modelu ne koristimo nasljeđivanje niti apstraktne klase, stoga ovaj princip nije direktno primijenjen. Sve klase su samostalne i ne postoji hijerarhija u kojoj bi se mogla izvršiti zamjena objekata. Liskov princip nije primijenjen, ali sistem nije ni dizajniran na način koji ga krši.

I – Interface Segregation Principle

Ovaj princip nalaže da se interfejsi trebaju dijeliti na manje, specifične cjeline. U ovom modelu interfejsi nisu razvijeni, stoga princip nije primijenjen. S obzirom da je sistem dizajniran jednostavno, bez interfejsa, nije došlo ni do njegovog kršenja.

D – Dependency Inversion Principle

Princip kaže da moduli visokog nivoa ne zavise od modula niskog nivoa, već obje trebaju zavisiti od apstrakcija. U ovom modelu nema zavisnosti preko interfejsa, niti je prisutna upotreba apstraktnih slojeva, pa se princip ne primjenjuje. Sve klase su konkretne i direktno referenciraju jedna drugu. Na primjer, **Dijete** referencira **Grupu** kao konkretan tip, a **Prisustvo** referencira **Dijete**, bez posredovanja apstrakcija.