Patterni ponasanja

- Strategy Pattern

Strategy dizajn patern izdvaja algoritam iz osnovne klase i omogućava njegovo dinamičko biranje putem različitih implementacija. U aplikaciji koja omogućava različitim korisnicima (studentima, profesorima, asistentima, adminima) da pregledaju statističke podatke, pristup tim statistikama se razlikuje:

- Student vidi broj svojih pitanja, komentara, odgovora i lajkova.
- Profesor vidi broj pitanja po predmetima koje predaje.
- Admin ima pristup sveobuhvatnoj statistici svih korisnika.

Korištenjem Strategy paterna, može se omogućiti biranje odgovarajuće strategije statistike bez dupliranja logike i bez izmjena glavnog kontrolera.

- State Pattern

State dizajn patern omogućava objektu da promijeni svoje ponašanje kada mu se promijeni stanje – kao da je riječ o drugom objektu. Svaki korisnik u sistemu ima status: aktivan ili neaktivan. Ovisno o tom statusu:

- Aktivan korisnik može dodavati pitanja, komentare i odgovore.
- Neaktivan korisnik ne može izvršavati nikakve interakcije – može samo gledati sadržaj.

Korištenjem State paterna, sistem jasno odvaja ponašanja u zavisnosti od statusa, čime se izbjegavaju stalne if (status == ...) provjere i povećava fleksibilnost za eventualna nova stanja (npr. banovani korisnik).

- Template Method Pattern

Template Method patern omogućava da se definira kostur algoritma u osnovnoj klasi, dok se pojedini koraci prepuštaju podklasama. Prilikom generisanja izvještaja o aktivnostima korisnika, svi izvještaji sadrže slične dijelove (broj pitanja, odgovora, komentara...), ali se način obrade i prikaza razlikuje:

- Student izvještaj je fokusiran na ličnu aktivnost.
- Profesor izvještaj prikazuje angažman studenata po predmetima.
- Admin izvještaj sadrži detaljnu analitiku sistema.

Primjenom ovog paterna omogućava se fleksibilno dodavanje novih tipova izvještaja bez izmjena osnovne logike.

- Observer Pattern

Observer patern definiše zavisnost između objekata tako da se promjene u jednom objektu automatski propagiraju ka drugim zainteresovanim objektima. U scenarijima gdje se korisničke akcije trebaju proširiti kao obavijest:

- Kada neko odgovori na pitanje, autor pitanja automatski dobija notifikaciju.
- Kada neko komentariše odgovor, autor odgovora bude obaviješten.
- Kada se objavi novi predmet ili zadatak, određeni korisnici mogu biti pretplaćeni na obavještenja.

Ovaj patern omogućava modularan i proširiv sistem obavještenja koji je lako povezati s drugim komponentama.

Command dizajn patern omogućava enkapsulaciju akcije kao objekta. Na taj način možeš fleksibilno upravljati izvršenjem komandi (npr. dodavanje, uređivanje, brisanje), kao i podržati mogućnost poništavanja (undo). Sistem ima različite radnje korisnika, posebno u vezi sa sadržajem:

- Dodavanje pitanja
- Lajkovanje odgovora
- Brisanje komentara
- Slanje poruke

Svaka od ovih radnji može biti modelirana kao komanda. Time bi sistem:

- Imao centralizovan način upravljanja akcijama
- Moguće je undo/redo budućih koraka (npr. greškom izbrisan komentar)
- Moguće je logovati svaku korisničku akciju

- Iterator Pattern

Iterator patern omogućava sekvencijalni prolazak kroz kolekciju objekata bez otkrivanja njene unutrašnje strukture. Kolekcije kao što su:

- Lista pitanja jednog korisnika
- Svi komentari ispod odgovora
- Sve poruke između dva korisnika
- Lista prijatelja

...često se prikazuju kroz pregledne liste, bez direktnog pristupa čitavom skupu u kontroleru.

Iterator omogućava:

- Lakše listanje sadržaja po pravilima
- Omogućava paginaciju, filtriranje, sortiranje uz minimalne promjene
- Jedinstven API za različite kolekcije

- Memento Pattern

Memento patern omogućava čuvanje prethodnih stanja objekta i njihovo vraćanje po potrebi – bez narušavanja enkapsulacije. Korisnici mogu uređivati sadržaj (npr. pitanja, odgovore). Zamislimo da student piše pitanje, ali:

- želi sačuvati verziju prije izmjene
- vrati se korak nazad (undo)
- uporedi verzije

Pitanje ili Odgovor objekat može imati memento objekat koji čuva ranije verzije, a korisnik ih vidi kroz historiju uređivanja.

Idealno i za buduće funkcije "Obnovi staru verziju".

- Mediator Pattern

Mediator patern centralizuje komunikaciju između objekata, tako da objekti ne komuniciraju direktno, već preko posrednika. U nasoj aplikaciji postoji razmjena poruka između korisnika:

- Student piše profesoru
- Asistent odgovara studentu
- Admin nadgleda komunikaciju

Umjesto da svaka klasa zna sve o drugim klasama, koristi se ChatMediator:

- Svaka poruka ide kroz jednog posrednika (PorukaService, GlobalniChat)
- Dodaju se lako pravila (filtriranje, moderacija, blok lista)

Posebno korisno za buduće proširenje na grupne razgovore, anonimne poruke ili notifikacije unutar poruka.