Patterni ponašanja

1. Strategy pattern

Strategy pattern odvaja algoritam iz glavne klase i smješta ga u posebne klase. Koristi se kada postoji više mogućih algoritama za rješavanje problema. U našem sistemu, funkcionalnost *best sellers* prikazuje najprodavanije proizvode na početnoj stranici i unutar kategorija. Lista se automatski generiše i ažurira prema broju prodatih proizvoda u posljednjih mjesec dana, što smo definisale kao kriterijum za rad algoritma. Strategy pattern možemo koristiti i pri sortiranju rezultata pretrage. Korisnik može pretraživati proizvode po nazivu ili cijeni. Rezultati su inicijalno sortirani rastuće, ali korisnik može izabrati i drugi način sortiranja prema svojim potrebama.

2. State pattern

State pattern omogućava objektima da mijenjaju ponašanje u zavisnosti od svog trenutnog stanja, koje se automatski mijenja prema pravilima sistema. To je dinamička varijanta Strategy patterna. U našem sistemu može se koristiti kada korisnik koji se registruje ali nije prijavljen, ima ista ograničenja kao gost – može samo pregledati proizvode i njihove detalje. Kupovina je omogućena tek nakon prijave.

3. Template method pattern

Template pattern služi za omogućavanje izmjene ponašanja u jednom ili više dijelova. Najčešće se koristi kada se za neki algoritam uvijek trebaju izvršiti isti koraci uz moguće izmjene za pojedinačne korake. Moguća primjena bi bila u Log in funkcionalnosti. Koraci odnosno potrebne informacije za prijavu su uvijek iste, kao i izgled stranice koji se otvori nakon prijave. Međutim, prilikom prve prijave nakon kreiranja računa, mogle bismo korisniku poslati email dobrodošlice ili mu poruku prikazati na ekranu.

4. Observer pattern

Observer pattern povezuje objekte tako da se promjene stanja jednog objekta automatski prenose na sve povezane. U našem sistemu, može se primijeniti za slanje notifikacija korisnicima koji nisu kupovali duže vrijeme, kao i za obavještenja o sniženjima, specijalnim ponudama ili kodovima za popust.

5. Iterator pattern

Ovaj pattern omogućava sekvencijalni pristup elementima kolekcije bez poznavanja kako je kolekcija struktuirana. Potencijalna ideja za implementaciju može biti da uvedemo opciju pregleda best seller proizvoda prema prilici ili prema cijeni. Za to bi nam bile potrebne dvije klase, Ocassionlterator i Pricelterator koje ce imati metodu nextProduct(): Product. Klase bi nasljeđivale interface IKreatorlterator sa metodama za kreiranje iteratora.

6. Chain of Responsibility pattern

Ovaj pattern omogućava da se složen proces obradi kroz lanac objekata, gdje svaki obrađuje jedan dio zadatka. U našem sistemu primjenjuje se pri realizaciji narudžbe – od pregleda i odabira proizvoda, dodavanja u korpu, unosa koda za popust, izbora načina plaćanja i preuzimanja, do slanja potvrde o kupovini putem emaila. Svaki korak obrađuju različite klase i kontroleri, čime se postiže bolja organizacija i fleksibilnost procesa.

7. Mediator pattern

Mediator pattern upravlja komunikacijom između objekata kako bi se izbjegla njihova direktna povezanost. Koristan je u sistemima sa velikim brojem međusobno povezanih klasa. U našem slučaju, zbog jednostavne strukture i manjeg broja veza, ovaj pattern trenutno nije primjenjiv.