PATERNI PONAŠANJA

1. Strategy pattern

Strategy patern izdvaja algoritme u zasebne klase i omogućava njihovu zamjenu u toku izvršavanja.

U našoj aplikaciji koristi se za sortiranje liste dostupnih vozača. Korisnik može izabrati da sortira rezultate prema cijeni, prosječnoj ocjeni ili vremenu dolaska. Svaka od ovih strategija sortiranja implementirana je kao zasebna klasa, dok aplikacija dinamički bira aktivnu strategiju na osnovu korisničkog izbora.

2. State pattern

State patern omogućava da objekat mijenja ponašanje zavisno od svog stanja.

U aplikaciji se koristi za klasu Voznja, koja može biti u stanju „na čekanju“, „prihvaćena“, „odbijena“ ili „završena“. Na osnovu trenutnog stanja vožnje, korisnicima i vozačima se prikazuju različite funkcionalnosti (npr. otkazivanje, ocjenjivanje, pregled detalja).

3. Template Method pattern

Template Method patern definiše osnovni tok algoritma, ali dopušta podklasama da redefinišu određene korake.

U aplikaciji se koristi tokom registracije korisnika. Svi korisnici prolaze osnovne korake (unos podataka, validacija, snimanje), ali različiti tipovi korisnika (putnik, vozač) mogu imati specifične dodatke kao što su prikaz dobrodošlice, pravila ponašanja ili obavezna dokumentacija za vozače.

4. Observer pattern

Observer patern omogućava obavještavanje svih zainteresovanih objekata kada se stanje nekog objekta promijeni.

U aplikaciji se koristi za obavještavanje korisnika. Kada vozač prihvati ili odbije vožnju, putnik automatski dobija obavijest. Takođe, kada se status vožnje promijeni (npr. vožnja završena), sistem šalje obavijesti putniku i omogućava ocjenjivanje. Ovaj patern može doprinijeti i automatizovanom marketingu kroz obavijesti o dostupnim vozačima.

5. Iterator pattern

Iterator patern omogućava prolazak kroz kolekciju podataka bez otkrivanja unutrašnje strukture.

U aplikaciji se koristi za prikaz rezultata pretrage vozača. Na osnovu filtera koji korisnik odabere (cijena, ocjena, udaljenost), koristi se odgovarajući iterator koji prolazi kroz rezultate i prikazuje ih na odgovarajući način.

6. Chain of Responsibility pattern

Chain of Responsibility patern omogućava obradu zahtjeva kroz lanac objekata.

U aplikaciji se koristi kod naručivanja vožnje. Prvo se provjerava da li su unesene lokacije ispravne, zatim se traže dostupni vozači, šalje se zahtjev, čeka se potvrda vozača, te na kraju dolazi do finalne potvrde ili otkazivanja. Svaki korak može samostalno odlučiti da nastavi lanac ili da ga prekine ako dođe do greške.

7. Mediator pattern

Mediator patern omogućava centralizovanu komunikaciju između objekata kako bi se smanjila njihova međusobna sprega.

U aplikaciji se koristi u komunikaciji između korisnika i administratora. U slučaju greške tokom rezervacije, plaćanja ili drugih problema, mediator posreduje između korisnika i sistema za podršku. Tako se osigurava da se poruke filtriraju, organizuju i obrađuju bez direktnog povezivanja svih klasa.