# Strukturalni patterni

## Adapter pattern:

Osnovna namjena Adapter paterna je da omogući širu upotrebu već postojećih klasa. U situacijama kada je potreban drugačiji interfejs već postojeće klase, a ne želimo mijenjati postojeću klasu koristi se Adapter patern. Adapter patern kreira novu adapter klasu koja služi kao posrednik između originalne klase i željenog interfejsa. Tim postupkom se dobija željena funkcionalnost bez izmjena na originalnoj klasi i bez ugrožavanja integriteta cijele aplikacije.

## Moguća primjena adapter patterna:

U našem slučaju adapter pattern bi se mogao koristiti da se omogući otvaranje naše stranice unutar neke druge, ili otvaranje dijelova nekih drugih stranica (npr. YouTube videa) na našoj stranici.

## Facade pattern:

Facade patern se koristi kada sistem ima više podsistema (subsystems) pri čemu su apstrakcije i implementacije podsistema usko povezane. Osnovna namjena Facade paterna je da osigura više pogleda (interfejsa) visokog nivoa na podsisteme čija implementacija je skrivena od korisnika. Operacije koje su potrebne određenoj korisničkoj perspektivi mogu biti sastavljene od različitih dijelova podsistema. Može se više fasada postaviti oko postojećeg skupa podsistema i na taj način formirati više prilagođenih pogleda na sistem. To omogućava jednostavnije korištenje sistema.

## Moguća primjena facade patterna:

U našem slučaju façade pattern bi se mogao primjeniti pri plaćanju kursa. Korisniku bi se prikazao ekran za popunjavanje podataka, dok bi se u

pozadini vršila obrada podataka, naplata kursa, te slanje maila za obavještenje o kupovini kursa.

## Decorater pattern:

Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Objekat pri tome ne zna da je urađena dekoracija što je veoma korisno za iskoristljivost i ponovnu upotrebu komponenti softverskog sistema. Decorator patern se koristi i kada postojeće klase komponenti nisu podesne za podklase, npr. nisu raspoložive ili bi rezultiralo u mnogo podklasa. Patern tretira komponente i dekoracije na potpuno isti način. Patern je idealan ako se želi primijeniti slojevito poboljšanje operacija objekta.

## Moguća primjena facade patterna:

U našem slučaju dekorater bi se mogao dodati ako bismo željeli da mjeri broj pregleda nekog kursa, tj. klikova na određeni kurs. Dodali bismo dekorater koji bi inkrementirao vrijednost na svaki klik na određeni kurs, što bismo mogli dalje iskoristiti za analitiku.

## Bridge pattern:

Osnovna namjena Bridge paterna je da omogući odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije. Kombinira 2 ili više ortogonalne hijerarhije klasa. Povezuje implementaciju za klasu za vrijeme izvršavanja. Dijeli implementaciju između više objekata.

Sprječava rapidno povećanja broja  klasa koje proizlaze iz interfejsa povezanog sa  puno implementacija.

## Moguća primjena bridge patterna:

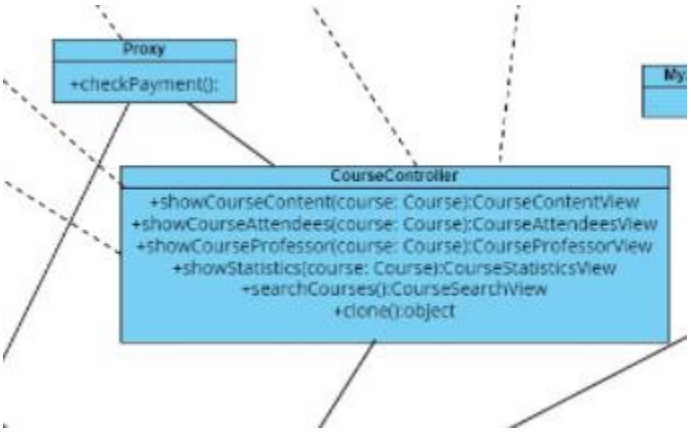
U našem slučaju bridge pattern bismo mogli iskoristiti pri logiranju i registrovanju korisnika. Bridgem bismo splojili usera i klasu za navedene akcije, pošto se oba korisnika (profesor i student) moraju registrovati i logirati.

## Proxy pattern:

Postoji više vrsta proxy paterna. Virtualeni proxy: Javni surogat objekat koji predstavlja kompleksni objekat čija aktivizacija se postiže na osnovu postavljenih pravila i u zadnjem mogućem trenutku. Remote proxy: Rješava probleme kada se objekt ne može  instancirati direktno. Omogućuje da se napravi poziv metode na objekat koji živi u unutrašnjosti različitih aplikacijskih domena ili servera. Zaštitni proxy: Omogućava pristup i kontrolu pristupa stvarnim objektima.

## Primjena proxy patterna:

U našem slučaju proxy pattern smo primjenili pri pokušaju ulaska na kurs koji se plaća. Kreirana je klasa proxy koja provjerava da li je kurs, koji se plaća, plaćen, ako jeste onda se dozvoljava prikaz, ako nije onda se korisnik upućuje na billing prikaz da plati kurs koji je odabrao.




## Composite pattern:

Osnovna namjena Composite paterna (kompozitni patern) je  da omogući formiranje strukture drveta pomoću klasa, u kojoj se individualni objekti (listovi stabla) i kompozicije individualnih objekata (korijeni stabla) jednako tretiraju.

## Moguća primjena composite patterna:

U našem slučaju mogli bismo iskoristiti composite pattern ako bismo željeli da imamo prikaz podataka nekog korisnika, nevažno da li je profesor ili student. Tako da bi composite se odnosio na User koji bi dalje odnosio na korisnika shodno koja je korisnik vrsta (profesor ili student).

## Flyweight pattern:

Flyweight patern treba da omogući da se svaki objekat na zahtjev  može dobaviti brzo. Postoje situacije u kojima je potrebno da se omogući razlikovanje dijela klase koji je uvijek isti za sve određene objekte te klase (intrinsic state-unutrašnje stanje) od dijela klase koji nije uvijek isti za sve objekte te klase (extrinsic state-vanjsko stanje). Optimizacijska tehnika koja omogućuje korištenje manje memorije eksternim pohranjivanjem podataka povezanih sa sličnim objektima.

# Primjena flyweight patterna:

U našem slučaju flyweight pattern smo iskoristili za kurseve koji se dijele na plaćene i besplatne. Tako da se akcije za obje vrste kurseva izvršavaju preko jedne klase User.