Univerzitet „Džemal Bijedić“ u Mostaru
Fakultet informacijskih tehnologija

SEMINARSKI RAD

Predmet: Formalne metode

PROFESORI:
prof. dr. Bernadin Ibrahimpašić

STUDENTI:
Kerim Begić IB220048
Elmir Mujkić IB220109
Kadir Mušija IB220334

MOSTAR, decembar, 2025.god.

1. Tehnike testiranja

Stranica na kojoj ćemo raditi tenike testiranja, automatizaciju testova i pronalaženje bugova je web shop https://elitnutrition.ba/.

S obzirom da za pojedine tehnike nećemo moći koristiti uvijek istu funkcionalnost, naglasit ćemo koju funkcionalnost tačno radimo prilikom svake tehnike testiranja.

1.1. Ekvivalentnost particioniranja

Funkcionalnost: Odabir količine prije dodavanja u korpu

U ovoj tehnici možemo posmatrati bilo koji proizvod koji se nalazi na web shopu.

Cilj ove tehnike je kreirati sve particije ekvivalencije (važeče i nevažeće) za određeni paramtar (količina), te za svaku particiju testirati po jednu vrijednost.

Definisali smo sljedeće particije;

- Sve vrijednosti <1
- Sve vrijednosti >=1
- Unos stringa

Nakon definisanja partricija uzimamo po jednu vrijednost iz svake particije za testiranje.

Parametar	Particija ekvivalencije	Ulazna vrijednost	Očeivani rezultat
količina	količina < 1	0	Poruka o nevažećoj vrijednosti
količina	količina >= 1	3	Dodavanje 3 proizvoda u korpu
količina	string	a	Nemogućnost unosa

1.1. Tabela ekvivalentnosti particioniranja

1.2. Analiza granične vrijednosti

Funkcionalnost: Filtiranje cijene

U ovoj tehnici posmatrali smo slider za filtriranje cijene na https://elitnutrition.ba/?orderby=date&product_cat=novo&s=&post_type=product (raspon 0–360 KM).

Cilj je bio da identifikujemo granične vrijednosti tog raspona i provjerimo ponašanje sistema tačno na granicama i neposredno oko njih, jer se tu greške najčešće pojavljuju.

Zbog toga smo definisali;

- particije ekvivalencije (cijena < 0 KM, 0–360 KM, cijena > 360 KM)
- granične vrijednosti (−1 KM, 0 KM, 1 KM, 359 KM, 360 KM, 361 KM).

Ove vrijednosti ćemo koristiti u testnim slučajevima podešavanjem slidera na konkretne iznose i provjeravati da li filtriranje vraća ispravne rezultate.

Parametar	Particija ekvivalencije	Granične vrijednosti za testiranje
cijena	cijena < 0 KM	cijena > 360 KM
cijena	0 KM ≤ cijena ≤ 360 KM	0 KM, 1 KM, 359 KM, 360 KM
cijena	cijena > 360	361 KM

1.2. Tabela analize graničnih vrijednosti

1.3. Testiranje tabele odluka (Decision Table Testing)

Funkcionalnost: Dodavanje proizvoda u korpu

U ovoj tehnici možemo posmatrati bilo koji proizvod koji se nalazi na web shopu.

Cilj ove tehnike bi bio preko logičkih vrijednosti (boolean) pokazati kombinacije uslova koji rezultiraju izvršavanjem radnji povezanih s tim pravilom

U nastavku slijedi tabela koja pokazuje funkcionalnost dodavanja u korpu koja treba zadovoljit uslov pakovanje+okus+količina da bi proizvod dodao u korpu.

USLOV	1	2	3	4	5	6	7	8
Pakovanje odabrano	T	T	T	T	F	F	F	F
Okus odabran	T	T	F	F	T	T	F	F
Količina odabrana	T	F	T	F	T	F	T	F
RADNJA								
Dodaj u korpu	T	F	F	F	F	F	F	F
Greška: odaberi pakovanje	F	F	F	F	T	T	T	T
Greška: odaberi okus	F	F	T	T	F	F	T	T
Greška: količina min 1	F	T	F	T	F	T	F	T

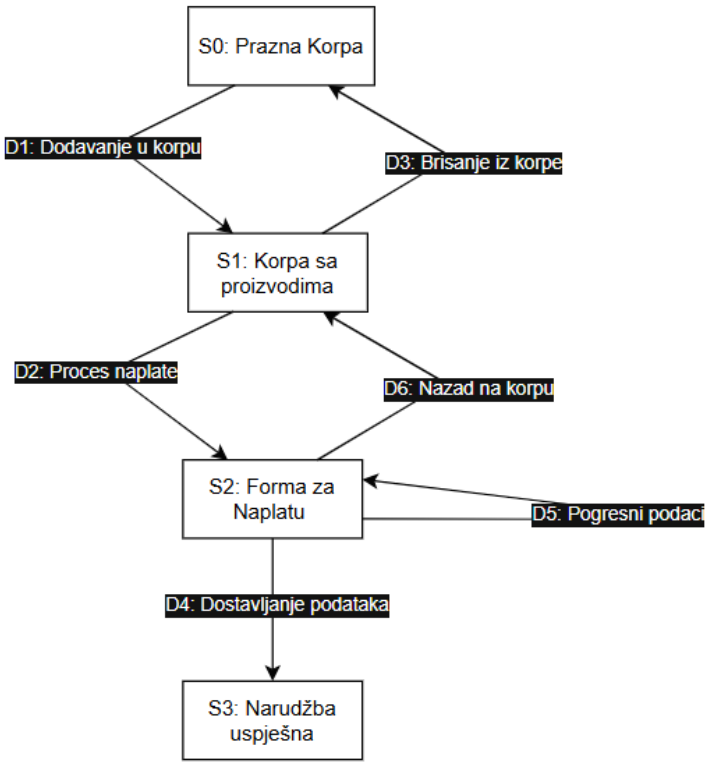
1.3. Tabela odluka

1.4. Testiranje tranzicije stanja

Funkcionalnost: Proces kupovine

Kod testiranja tranzicije stanja posmatrali smo proces kupovine na https://elitnutrition.ba/cart/ kao niz stanja kroz koja korisnik prolazi (korpa prazna, korpa s proizvodima, checkout forma, potvrđena narudžba).

Cilj je bio da provjerimo da li svaka akcija korisnika dovodi sistem u ispravno sljedeće stanje.



1.4 Dijagram tranzicije stanja

Testni slučajevi

- TC-041: Dodavanje u praznu korpu
- TC-042: Brisanje iz korpe
- TC-043: Uspješni proces narudžbe
- TC-044: Nevalidni podaci pri procesu naplaćivanja
- TC-045: Povratak na korpu – zadržavanje stavki

1.5. Testiranje slučajeva upotrebe

Funkcionalnost: Pretraživanje proizvoda

U ovoj tehnici posmatrali smo funkcionalnost pretraživanja koja se nalazi na glavnom navigacionom baru.

Cilj ove tehnike bio je kreirati slučajeve upotrebe, svaki slučaj treba imati preduslove koje treba ispuniti za uspješan rad.

Za demonstraciju ove tehnike možemo pogledati kod koji prikazuje različite slučajeve upotrebe i tabelu u kojoj je svaki slučaj obrađen.

```
function searchProducts(query):
1  query = trim(query)

2  if query == "":
3     show("Enter a search term")
4     return

5  if containsSqlInjectionIndicators(query):
6   show("Invalid search term")
7   return

8 results = findProducts(query)

9 if results.count == 0:
10    show("No results found")
11     return

12 renderResults(results)
```

TEST CASE	Search term	Results exist	Expected result	Covered lines
TC1	(empty) or " "	No (empty)	Error: "Enter a search term" (search not executed).	1,2,3,4
TC2	asfagsdgh	No	Message: "No results found" (empty results state).	1,2,5,8,9,10,11
TC3	whey	Yes	Results are displayed.	1,2,5,8,9,12
TC4 (SQLi)	' (single quote)	N/A	Invalid search term; no crash/500; no SQL error details shown.	1,2,5,6,7
TC5 (SQLi)	' OR '1'='1	N/A	Invalid search term; no SQL errors; no unexpected "all products" bypass.	1,2,5,6,7

1.6. Testiranje odluka (decisions) i pokrivenost

Funkcionalnost: Proces kupovine

Kod testiranja odluka pratili smo logiku obračuna dostave i popusta kroz niz if uslova u pseudokodu.
Cilj je bio da svaka ključna odluka bude barem jednom pokrivena testovima: da li je korpa prazna (cart_items_count == 0), da li je total iznad ili jednako 100 KM (total >= 100), te da li kupon postoji i da li je validan (coupon != null, coupon.valid == true).

Definisane su varijable **can_checkout**, **shipping_fee**, **discount** i **final_price**, uz posebnu akciju **CLEAR_CART** koja postavlja cart_items_count i total na 0. Na osnovu uslova određuju se can_checkout, iznos dostave i popust, a zatim se računa final_price = total + shipping_fee - discount, što omogućava provjeru efekta svake odluke kroz različite testne slučajeve.

```
if (action == "CLEAR_CART") {
   cart_items_count = 0;
   total = 0;
}

if (cart_items_count == 0) {
   can_checkout = false;
   shipping_fee = 0;
   discount = 0;
} else {

   can_checkout = true;

   if (total > 100) {
      shipping_fee = 7;
   } else {
      shipping_fee =  0;
   }

   if (coupon != null) {
      if (coupon.valid == true) {
         discount = 10;
      } else {
         discount = 0;
      }
   } else {
      discount = 0;
   }
}

final_price = total + shipping_fee - discount;
```

1.6. Pseudokod za testiranje odluka i pokrivenost

Testni slučajevi

- TC-061: Korpa puna, total < 100 KM, bez kupona
- TC-062: Korpa puna, total >= 100 KM, bez kupona
- TC-063: Korpa puna, total > 100 KM, kupon nevalidan
- TC-064: Ažuiriranje korpe dodavanjem
- TC-064: Ažuiriranje korpe brisanjem
- TC-066: Brisanje sve iz korpe

1.7. Pogađanje pogreške

Funkcionalnost: Dodavanje proizvoda u korpu

U ovoj tehnici možemo posmatrati bilo koji proizvod koji se nalazi na web shopu.

Cilj ove tehnike je na osnovu iskustva predvidjeti defekte te nabrojati popis mogućih defekata.

No	Potencijalna greška	Testni slučaj
1	"Dodaj u korpu" radi bez odabranog pakovanja.	Pokušaj dodati proizvod u korpu bez odabira pakovanja (okus odabran, qty=1) klikom na "Dodaj u korpu".
2	"Dodaj u korpu" radi bez odabranog okusa.	Pokušaj dodati proizvod u korpu bez odabira okusa (pakovanje odabrano, qty=1) klikom na "Dodaj u korpu".
3	Dozvoljena količina 0.	Pokušaj postaviti količinu na 0 (manuelni unos ako je moguće) i klikni "Dodaj u korpu".
4	Dodaje se pogrešan okus (UI prikazuje jedno, u korpu ode drugo).	Odaberi konkretan okus (npr. "Whey"), qty=1; klikni "Dodaj u korpu"; otvori korpu i provjeri da li je isti okus.
5	Dodaje se pogrešno pakovanje (varijanta mismatch).	Odaberi konkretno pakovanje (npr. 420g), odaberi okus, qty=1; klikni "Dodaj u korpu"; otvori korpu i provjeri pakovanje.
6	Količina se ne prenese (uvijek doda 1).	Povećaj qty na 3; klikni "Dodaj u korpu"; otvori korpu i provjeri da li je qty=3.
7	Dupli klik doda stavku dva puta (race condition).	Brzo klikni 2× "Dodaj u korpu" za istu varijantu (qty=1) i provjeri da nema dupliranja/nekonzistentnosti u korpi.
8	Višestruko dodavanje iste varijante pravi duple redove umjesto povećanja qty (ili obratno).	Dodaj istu varijantu u korpu dva puta i provjeri da li se ponaša konzistentno (merge qty ili novi red, prema pravilima).
9	Greška servera/timeout pokaže "Dodano" iako nije dodano.	Simuliraj offline/timeout/500; klikni "Dodaj u korpu"; provjeri da se ne prikazuje lažna potvrda i da korpa ostaje nepromijenjena.
10	Refresh resetuje izbor, ali dugme i dalje dozvoljava dodavanje bez ponovnog odabira.	Odaberi pakovanje+okus; refresh; bez ponovnog odabira klikni "Dodaj u korpu" i provjeri da validacija spriječi dodavanje.

1.7. Popis mogućih defekata

1.8. Istraživačko testiranje

Funkcionalnost: Proces kupovine

Istraživačko testiranje funkcionalnosti na https://elitnutrition.ba/ organizovali bismo kao session-based exploratory testing koristeći **RCRCRC**.

Kod **Recent** dijela bavili bismo se elementima koji djeluju najnovije ili najdinamičnije, prije svega filterom po cijeni (slider 0–360 KM) i prikazom akcijskih cijena. Testirali bismo različite opsege filtera (0–100, 100–200, 0–360), kombinacije sa drugim filterima i ponašanje sistema pri dodavanju i uklanjanju proizvoda iz korpe, uz provjeru ažuriranja ukupne vrijednosti.

U okviru **Core** segmenta fokusirali bismo se na osnovne tokove: dodavanje jednog proizvoda u korpu, prolazak kroz cijeli checkout sa ispravnim podacima i uspješnu potvrdu narudžbe. Zatim bismo dodali scenarije sa više različitih proizvoda i količina da provjerimo da li se ukupna cijena i eventualna dostava ispravno računaju.

Za **Risky** dio bavili bismo se dijelovima koji nose veći rizik za ozbiljne greške: granice filtera cijene (0, 1, 359, 360), unos vrlo dugih stringova i specijalnih znakova u polja za ime, adresu i napomenu, neispravne e-mail adrese i rad sa kupon kodovima, ako postoje. Testirali bismo i osvježavanje stranice tokom checkouta, potencijalni istek sesije i brzo višestruko klikanje na dugme za potvrdu narudžbe, jer takve situacije često otkrivaju kritične bugove.

U **Configuration** fazi iste tokove prolazili bismo u različitim browserima (Chrome, Firefox, Edge) i na mobilnom uređaju, kako bismo provjerili da li se UI, filteri i checkout ponašaju konzistentno. Ako aplikacija podržava i gosta i prijavljenog korisnika, testirali bismo oba scenarija zbog mogućih problema sa učitavanjem i spremanjem podataka u checkoutu.

Za **Repaired** dio fokusirali bismo se na područja gdje su ranije prijavljeni bugovi (npr. netačan izračun total cijene ili problemi sa filterom/kuponom) i ponovo bismo izvršavali iste scenarije uz nekoliko varijacija da potvrdimo da su ispravke uspješne.

Na kraju, u **Chronic** dijelu bavili bismo se potencijalno problematičnim zonama kao što su korpa i checkout, ponavljajući tipične radnje: više puta dodavanje i uklanjanje proizvoda, promjena količina, korištenje back/forward dugmadi u browseru i reload stranice u različitim fazama checkouta. Time bismo pokušali otkriti hronične probleme poput nestanka artikala iz korpe, nekonzistentnih total iznosa ili kreiranja duplih narudžbi.