**SE321 OBEZBEĐENJE KVALITETA, TESTIRANJE I ODRŽAVANJE**

**SOFTVERA**



**Jesenji semester** **2022/23**

**PROJEKTNI ZADATAK**

PROJEKAT- Sistem za rad hotela

Student: Bojana Stajić 4596

Asistent: Sara Nikolić

Sadržaj projekta:

[Uvod 3](#_Toc124342482)

[Specifikacija softverskog sistema koji se testira (fokus je na zahtevima softverskog sistema) 3](#_Toc124342483)

[Kratak opis poslovnog problema koji se rešava softverskim sistemom i njegova veza sa okruženjem 4](#_Toc124342484)

[Funkcionalni zahtevi softverskog sistema 4](#_Toc124342485)

[Nefunkcionalni zahtevi softverskog sistema 5](#_Toc124342486)

[Dijagram slučajeva korišćenja softverskog sistema 5](#_Toc124342487)

[Funkcionalnosti softverskog sistema koje će biti testirane 6](#_Toc124342488)

[Funkcionalnosti softveskog sistema koje neće biti testirane 7](#_Toc124342489)

[Strategija testiranja 7](#_Toc124342490)

[Tipovi testiranja koji će se sprovesti i u kojoj meri 7](#_Toc124342491)

[Kriterijumi 7](#_Toc124342492)

[Tablica povezanosti zahtevi-slučajevi testiranja 8](#_Toc124342493)

[Osoblje uključeno u testiranje 8](#_Toc124342494)

[Vremenski raspored testiranja 9](#_Toc124342495)

[Primena 12 Casper pravila 9](#_Toc124342496)

[Realizacija testiranja softverskog sistema 10](#_Toc124342497)

[Pregled koda 10](#_Toc124342498)

[Primena McCabe-ova siklomatske složenosti 11](#_Toc124342499)

[*Test slučaj 1* 11](#_Toc124342500)

[*Test slučaj 2* 12](#_Toc124342501)

[*Test slučaj 3* 13](#_Toc124342502)

[JUnit testiranje 14](#_Toc124342503)

[Testiranje metodom crne kutije – Black Box 16](#_Toc124342504)

[Ekvivalentno parcelisanje 16](#_Toc124342505)

[Testiranje metodom bele kutije – White Box 17](#_Toc124342506)

[Pokrivenost petlji 18](#_Toc124342507)

[Integraciono testiranje 18](#_Toc124342508)

[Izveštaj o testiranju 19](#_Toc124342509)

[ODRŽAVANJE, DALJI RAZVOJ I KONTINUALNI PROCES POBOLJŠANJA SOFTVERA 20](#_Toc124342510)

[Model estimacije troškova prilikom održavanja odabrane aplikacije 20](#_Toc124342511)

[Pristup za postizanje višekratne upotrebljivosti 21](#_Toc124342512)

[ZAKLJUČAK 22](#_Toc124342513)

[LITERATURA 23](#_Toc124342514)

# Uvod

Testiranje softvera predstavlja krucijalnu fazu u procesu kreiranja jednog naprednog I funkcionalnog softvera, a u isto vreme i neophodna, kako bi se otkrile greške načinjene u svim fazama razvoja softvera, amogu se ispraviti nakon otkrivanja. Testiranje softvera obuhvata različite vrste testova kako bi se utvrdilo da softverski proizvod neće imati funkcionalne i nefunkcionalne nedostatke, naravno, sve u cilju smanjenja celokupnih troškova razvoja samog softvera, kao i poboljšanja njegovog kvaliteta. Ukoliko softver ne izvršava funkciju za koju je namenjen, on je neefikasan, i samim tim će biti zamenjen, i postaće neprofitablian.

Softversko testiranje predstavlja aktivnost koja obuhvata kompletan process razvoja I održavanja, i time ga čini bitnim delom celokupne konstrukcije softvera. Testiranje nije samo aktivnost koja se praktikuje nakon implementiranja faze kodiranja, već je aktivnost koja se izvodi zbog evaluacije kvaliteta proizvoda i njegovog stalnog poboljšanja, putem identifikacije problema i nedostataka samog sistema.

# Specifikacija softverskog sistema koji se testira (fokus je na zahtevima softverskog sistema)

## Kratak opis poslovnog problema koji se rešava softverskim sistemom i njegova veza sa okruženjem

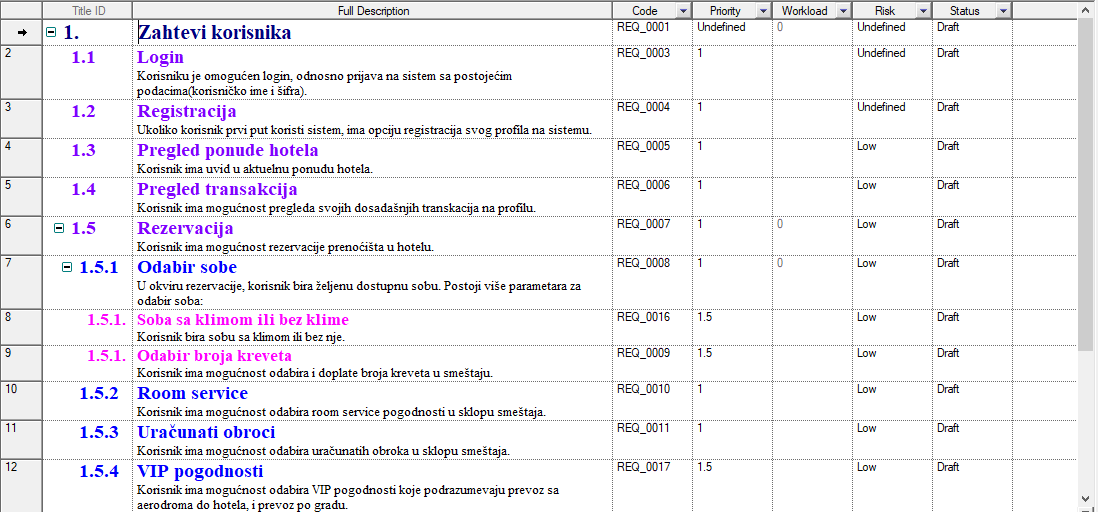
Aplikacija za predlog za projektni zadatk jeste sistem koji će olakšati rad jednog hotela, kao i njegovih zaposlenih i mušterija. Ideja je da korisnik ima uvid u ponudu hotela, kao i u svoje transakcije u istom, kao i mogućnost odabira soba, broja kreveta, room service-a, uračunatih obroka i tako dalje. Sistem omogućava zaposlenim i korisnicima da se uloguju putem korisničkog imena i šifre , a ukoliko nemaju profil, pristupaju procesu registracije.

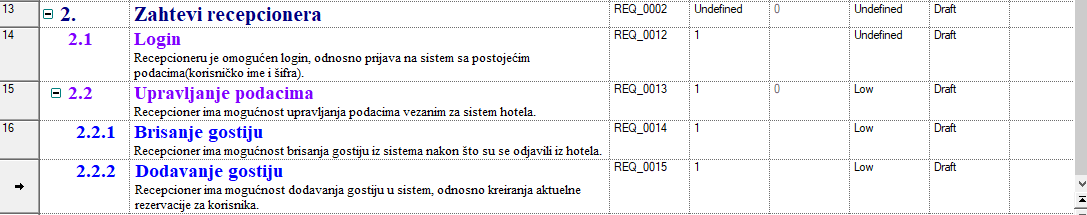
Tipovi korisnika su recepcioner i korisnik. Recepcioner ima mogućnost dodavanja gostiju u bazu podataka, raspoređivanje po sobama, odnosno kreiranje rezervacije specifičnog gosta. Recepcioner takođe može da briše goste iz sistema nakon što napuste hotel. Unutar hotela postoji više tipova soba: specifične po broju kreveta, sa klimom ili bez.

Običan korisnik ima mogućnost uvida u ponudu hotela, kao i odabira soba i njene rezervacije kao i opcionalno neke dodatne mogućnosti i pogodnosti koje želi da kupi.

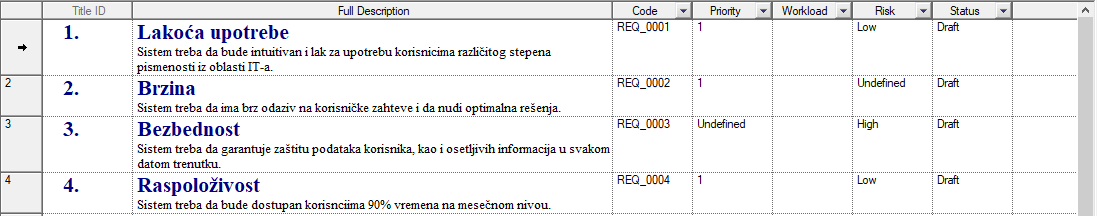
Kod običnog korisnika postoji VIP članstvo, koje podrazumeva prevoz sa aerodroma do hotela, i prevoz po gradu.

## Funkcionalni zahtevi softverskog sistema

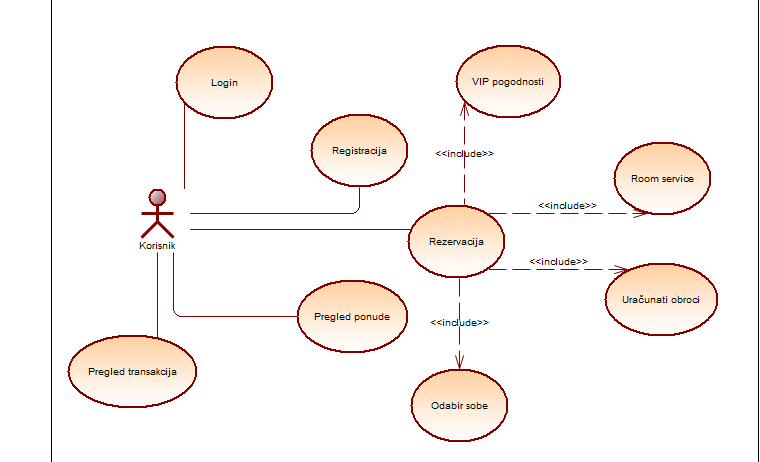


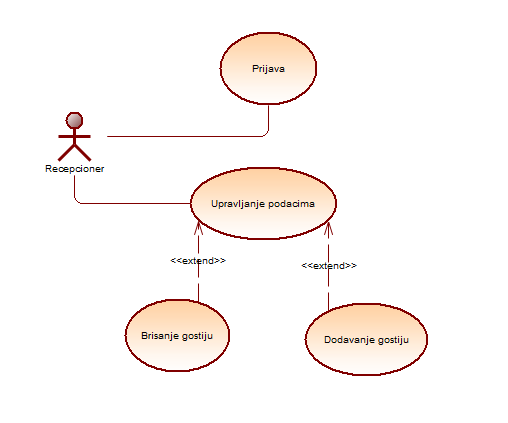


## Nefunkcionalni zahtevi softverskog sistema



## Dijagram slučajeva korišćenja softverskog sistema





## Funkcionalnosti softverskog sistema koje će biti testirane

1. Baza podataka sa svim podacima vezanim za hotel i goste
2. Funkcionalnost pregleda ponuda
3. Funkcionalnost pregleda transakcija
4. Funkcionalnost rezervacije
5. Funkcionalnost registracije
6. Funkcionalnost prijave
7. Funkcionalnost upravljanja podacima

|  |  |  |  |
| --- | --- | --- | --- |
| Funkcionalnost | Izdanje 1.0 | Izdanje 1.1 | Izdanje 1.2 |
| FE-1 | Potpuno implementirano | / | / |
| FE-2 | Potpuno implementirano | / | / |
| FE-3 | Nije implementirano | Potpuno implementirano | / |
| FE-4 | Potpuno implementirano | / | / |
| FE-5 | Potpuno implementirano | / | / |
| FE-6 | Potpuno implementirano | / | / |
| FE-7 | Nije implementirano | Potpuno implementirano | / |

## Funkcionalnosti softveskog sistema koje neće biti testirane

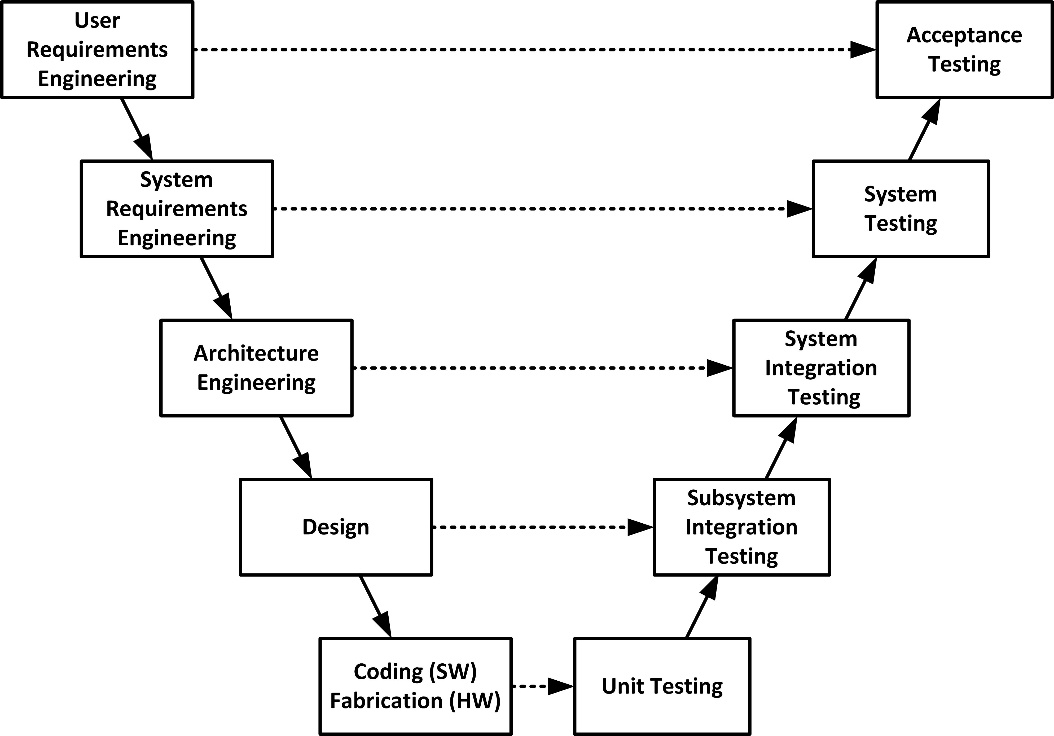
Sve funkcionalnosti će biti obuhvaćene tokom testiranja.

# Strategija testiranja

## Tipovi testiranja koji će se sprovesti i u kojoj meri

Kao i u slučaju vodopada, u pitanju je sekvencijalno izvršavanje procesa. Svaka faza mora biti kompletirana pre nego što počne sledeća. Testiranje proizvoda se planira paralelno sa odgovarajućom fazom razvoja u V modelu.

Zahtevi se analiziraju na početku modela. Pre nego što se krene sa razvojem, kreira se plan sistemskog testiranja. Ovaj plan testiranja se fokusira na specificirane funkcionalnosti koje su definisane u fazi prikupljanja zahteva.



### Kriterijumi

* Broj grešaka kategorije 1=0
* Broj grešaka kategorije 3=0
* Broj grešaka kategorije 2 manje od 5 po modulu;
* Broj uspešno obavljenih testova mora biti 99 %;
* Svi planirani testovi moraju biti izvršeni

## Tablica povezanosti zahtevi-slučajevi testiranja

Dat je primer tablice povezanosti za funkcionalnost koja će biti testirana u okviru projekata. Tablica povezanosti pokazuje koji testovi testiraju koju funkcionalnost. Na osnovu nje možemo utvrditi da li neki od testova nedostaju, ako vidimo da postoje funkcionalnosti bez testova.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Broj zahteva** | **Test 1-login** | **Test 2-Pregled ponuda** | **Test 3-Uvid u transakcije** | **Test 4-Rezervacija** | **Test 5-Upravljanje podacima** | **Test 6 –**  **Registracija** |
| **REQ\_1** | **X** | **X** | **X** | **X** | **X** |  |
| **REQ\_2** |  | **X** |  | **X** | **X** | **X** |
| **REQ\_3** | **X** |  | **X** |  |  | **X** |
| **REQ\_4** | **X** | **X** | **X** | **X** |  | **X** |
| **REQ\_5** |  |  |  |  |  | **X** |
| **REQ\_6** | **X** |  |  |  |  | **X** |
| **REQ\_7** | **X** | **X** | **X** |  | **X** |  |

## Osoblje uključeno u testiranje

|  |  |  |
| --- | --- | --- |
| Uloga | Korisnik | Testira |
| Tester 1 | Bojana Stajić | * Funkcionalnost pregleda ponuda * Funkcionalnost pregleda transakcija * Funkcionalnost upravljanja podacima |
| Tester 2 | Vladimir Mitrović | * Funkcionalnost registracije * Funkcionalnost prijave |
| Tester 3 | Marija Kostić | * Funkcionalnost rezervacije |

## Vremenski raspored testiranja

|  |  |  |
| --- | --- | --- |
| Aktivnost | Planirani rok | Krajni rok |
| Poručivanje sistema | 15.10.2022. | 15.10.2022. |
| Pisanje test slučaja | 25.10.2022. | 11.11.2022. |
| Izvršavanje test slučajeva | 12.11.2022. | 20.11.2022. |
| Isporuka | 10.01.2023. | 15.01.2023. |

## Primena 12 Casper pravila

1. **Estimacija veličine izvornog koda**

Broj FP= broj ulaza x 4 + broj izlaza x 5 + broj upita x 4 + broj matičnih datoteka x 10 + broj interfejsa x 7

FP = 5\*4 + 2\*5 + 3\*4 + 4\*10 + 2\*7 = 96

L= 96 x 53= 5088 LOC

1. **Estimacija dokumentacije**

Dokumentacija= FP 1.15 stranica

Dokumentacija= 1601.15 = 342 stranice

1. **Estimacija odudaranja korisničkih zahteva**

Troškovi odudaranja od korisničkih zahteva će biti na mesečnom nivou u proseku 3% vrednosti od celokupno planiranog budžeta.

1. **Estimacija broja slučajeva testiranja**

FP(broj podešenih funkcionalnih tačaka)=160

Broj test slučajeva= FP1.2

Broj test slučajeva=1601.2= 441 test slučajeva

1. **Estimacija potencijalnog broja grešaka**

Potencijalni broj grešaka=FP1.25

Potencijalni broj grešaka= 1601.25= 569 mogućih grešaka

1. **Estimacija efikasnosti otklanjanja greške**

U svakoj fazi testiranja broj pronađenih i otklonjenih grešaka je u proseku 20.

1. **Estimacija efikasnosti organizovanog otklanjanja grešaka**

Inspekcija dizajna će pronaći i otkloniti u proseku 80% grešaka u sistemu, ovaj pristup će biti skuplji od pretnodnog u koraku 6, ali će biti efikasniji.

1. **Estimacija efikasnosti otklanjanja grešaka nakon puštanja softvera u rad**

Programeri zaduženi za održavanje softvera mogu ispraviti određen broj grešaka koji zavisi od TMM i CMM nivoa.

1. **Estimacija trajanja realizacije projekta**

FP(broj podešenih funkcionalnih tačaka)=160 DM=FP0.4 [KM] kalendarskih meseciDM= FP0.4=1600.4= 7.5 [KM] kalendarskih meseci

**10. Estimacija potrebnih ljudi za realizaciju projekta**

FP(broj podešenih funkcionalnih tačaka)=160

Prosečna produktivnost projektanata=150

Broj projektanata= FP/Prosečna produktivnost projektanata Broj projektanata=160 /= 2 projektanta

**11. Estimacija ljudi potrebnih za održavanje softvera**

FP(broj podešenih funkcionalnih tačaka)=160

Broj ljudi za održavanje= FP/ prosečna efikasnost održavanja

Broj ljudi za održavanje=FP/ 750= 160 /750 = 1 čovek

**12. Estimacija ukupnih napora u realizaciji softverskog projekta**

Ukupni Napor= Ukupno vreme \* broj ljudi= 7.5\*2 = 15 čovek-meseci

# Realizacija testiranja softverskog sistema

## Pregled koda

Pregled koda je kod razvojnih timova sve češća i češća praksa u svrhu razvoja kvalitetnijeg koda. Pre puštanja koda u produkciju, kod najpre ide na proveru. Jedan od poznatijih primera ove prakse jeste platforma GitHub na kojoj programeri objavljuje svoje programe, kodove i projekte, u vidu repozitorijuma sa opisom svakog svog „commit-a“ pa zatim dobijaju predlog za korekciju i izmenu koda koju mogu da prihvate ili odbace.

## Primena McCabe-ova siklomatske složenosti

**v(G) = E - N + 2P**

• E - broj grana na grafu programa,

• N - broj čvorova na grafu,

• P - broj nepovezanih komponenti ili se može posmatrati kao broj izlaza iz programa

**v(G) = 9 – 8 + 2 = 3 test slučaja**

## Test slučaj 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera uračunatog obroka opcije | Rev 1 | Autor | Bojana Stajić | Datum | 12.11.2022. |
| Cilj | Provera ispravnosti sistema za rezervaciju | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 3 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da korisnik dobije opciju za uračunat obrok u okviru svoje rezervacije |
|  | Aplikacija je pokrenuta od strane korisnika koji bira opciju za uračunate obroke |
|  | Korisnik koristi računar za pristup sistemu |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je pristupio  I odabrao opciju Obroci | / | Nakon odabira opcije obroka sistem otvara korisniku dodatne opcije za obroke | Sistem je odreagovao očekivano i odradio naredbu | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Korisniku se prikazuju odredjene opcije za obroke u sklopu rezervacije |

## Test slučaj 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera uračunatog obroka opcije | Rev 1 | Autor | Bojana Stajić | Datum | 12.11.2022. |
| Cilj | Provera ispravnosti sistema za rezervaciju | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 3 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da korisnik dobije opciju za uračunat obrok u okviru svoje rezervacije |
|  | Aplikacija je pokrenuta od strane korisnika koji bira opciju za uračunate obroke |
|  | Korisnik koristi računar za pristup sistemu |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je pristupio  I odabrao opciju Obroci | / | Nakon odabira da rezerviše bez obroka korisniku se dalje prikazuje naredne stavke za rezervaciju | Sistem je odreagovao očekivano i odradio naredbu | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Korisniku se prikazuju izvreštaj za rezervaciju koja nema uračunat obrok. |

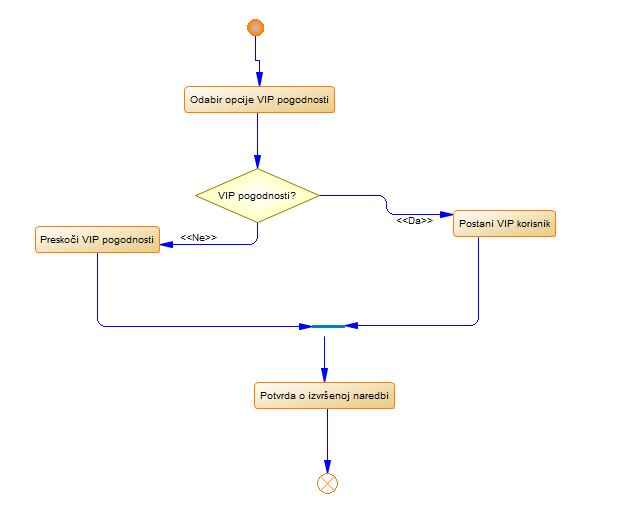
## Test slučaj 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Provera uračunatog obroka opcije | Rev 1 | Autor | Bojana Stajić | Datum | 12.11.2022. |
| Cilj | Provera ispravnosti sistema za rezervaciju | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 3 min |

|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Postavka aplikacije je takva da korisnik dobije opciju za uračunat obrok u okviru svoje rezervacije |
|  | Aplikacija je pokrenuta od strane korisnika koji bira opciju za uračunate obroke |
|  | Korisnik koristi računar za pristup sistemu |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je pristupio  i nije obeležio ni da želi ni da ne želi obroke. | / | Sistem obaveštava korisnika da polje za opciju obroka ne može biti nepopunjeno. | Sistem je odreagovao očekivano i izbacio poruku | 0 |

|  |  |
| --- | --- |
| Opis postuslova | |
|  | Korisniku se prikazuje sistemska poruka kako polje za obrok ne može ostati nepopunjeno. |



**v(G) = E - N + 2P**

• E - broj grana na grafu programa,

• N - broj čvorova na grafu,

• P - broj nepovezanih komponenti ili se može posmatrati kao broj izlaza iz programa

**v(G) = 8 – 7 + 2 = 3 test slučaja**

## JUnit testiranje

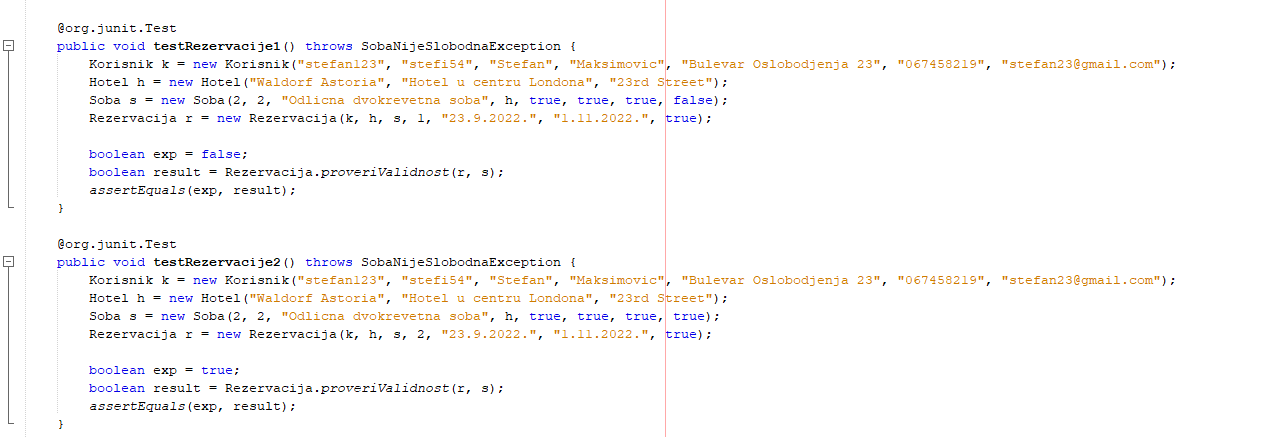
Testiranje modula proverava da li sve komponente i pojedinačne funkcije rade kako treba, odnosno sa određenim ulazima dobijaju se ispravni izlazi. Jedinično testiranje treba vršiti tamo gde rezultati metoda koja se testiraju mogu biti proučeni i mogu im se dodeljivati određene vrednosti u svrhu testiranja.

U sklopu Java programskog jezika postoji ugrađena opcija za testiranje, odnosno JUnit. To je framework za testiranje u Javi i kao takav lak je za upotrebu i generisanje. U slučaju projektnog zadatka, testiramo sve klase koje su generisane u Javi a napravljene su u PowerDesigner alatu.

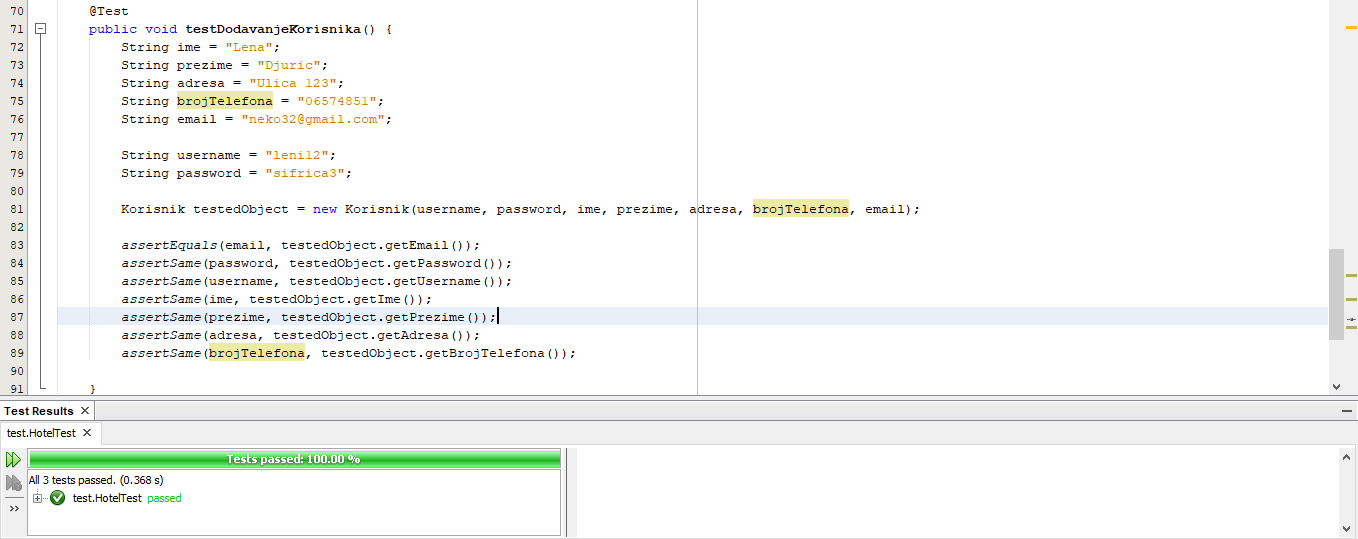
JUnit testovi koje smo uradili su za sledeće funkcionalnosti sistema:

* Rezervacija
* Dodavanje gostiju

U prikazanom slučaju, svi testovi prošli su bez grešaka.



Testove rezervacije sam vršila po sledećem konceptu: Unesem podatke o korisniku, hotelu i hotelskoj sobi koja treba da se rezerviše, a ta soba ima poslednju vrednost u konstruktoru tipa boolean koja prikazuje da li je ta soba slobodna ili ne. Dva test slučaja gore prikazana ispituju prvo kada soba nije slobodno da li se sistem ponaša na adekvatan način, odnosno mi očekujemo false vrednost koja ja tačna pa je test prošao uspešno. U drugom testu, soba je slobodna te za metod koji ispituje da li je rezervacija validna očekujemo true, što i dobijamo te je i ovaj test prošao uspešno.



Naredni test se bavi testiranjem dodavanja korisnika sistema, odnosno ispituje se konstruktor. Kao što vidimo na dnu ove slike, svi testovi su prošli uspešno.

## Testiranje metodom crne kutije – Black Box

### Ekvivalentno parcelisanje

U tehnici podele na klase ekvivalencije polazi se od ideje da se ulazni podaci mogu podeliti u reprezentativne klase tako da se za sve pripadnike jedne klase program ponaša na sličan način. Takve klase su nazvane klasama ekvivalencije. Klase ekvivalencije su međusobno disjunktne i pokrivaju ceo prostor vrednosti ulaza. Što se testiranja tiče, ono se obavlja samo za jednu reprezentativnu vrednost ulaza iz svake klase ekvivalencije , jer se u praksi smatra da je to jednako delotvorno kao i testiranje sa bilo kojom drugom vrednošću iz iste klase, te se štedi vreme i utrošeni resursi.

Dozvoljena vrednost za unos broja sobe je broj koji je prethodno korisnik mogao da vidi u ponudi, a da je slobodan, nedozvoljeni brojevi su negativni ili nepostojeći.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Naslov | Validnost unetog broja sobe za rezervaciju | Rev 1 | Autor | Bojana Stajić | Datum | 24.11.2022. |
| Cilj | Provera ispravnosti sistema za validaciju pri unosu broja sobe za rezervaciju | | Reference |  | | |
| Test uslovi |  | | | Neophodno vreme za izvršenje test slucaja | | 5 min |

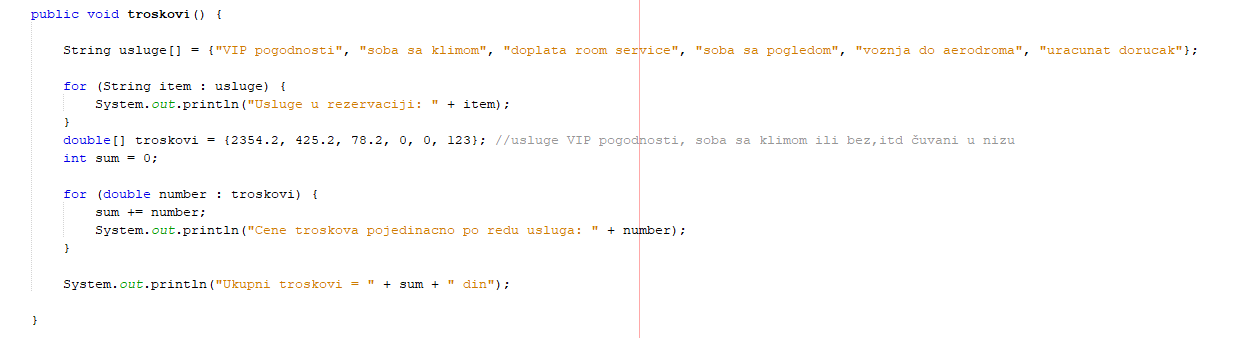
|  |  |
| --- | --- |
| Opis postavke za testiranje | |
|  | Sistem prihvata samo predefinisani odnosno dozvoljeni format broja , odnosno brojevi koji su dostupni, i ne brojevi koji su negativni ili 0 |
|  | Aplikacija je pokrenuta i to od korisnika |
|  | Korisnik koristi računar i tastaturu za unos podataka |

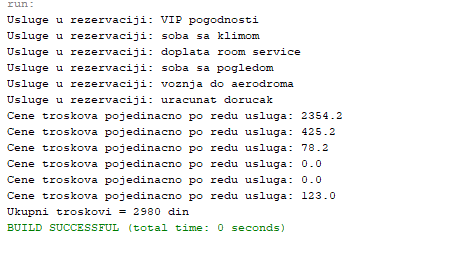
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Definicija testa | | | Izvršenje testa | |
| Uslovi | Ulazni podaci | Očekivani rezultati | Aktuelni rezultati | Broj problema |
| Korisnik je izvršio unos broja sobe za rezervaciju u hotelu. | Unet je broj 432 koji odgovara validnom broju sobe koja je slobodna. | Ukoliko uneti brojevi nisu korektni, sistem je dužan da izbaci grešku i o tome obavesti korisnika. | Sistem je odreagovao očekivano jer je broj koji je unet dobar, te je izvršio rezervaciju uspešno. | 0 |

## Testiranje metodom bele kutije – White Box

White Box metod testiranja ili strukturno testiranje podrazumeva ispitivanje interne strukture programa ili sistema. Tester koji testira poznaje internu strukturu i logiku programa. Testni podaci se dobijaju putem ispitivanja logike programa ili sistema, bez brige o zahtevima koje on treba da zadovolji.

## Pokrivenost petlji

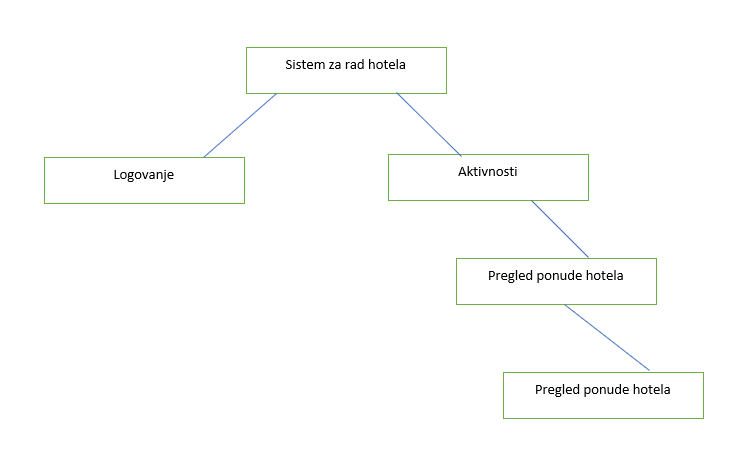




U gore prikazanom primeru, gledamo pokrivenost petlje u metodi gde petlja pretražuje parametre dok ne izlista sve troškove korisničke rezervacije. Postoje dva niza, jedan čuva moguće usluge, a drugi koliko je korisnik potrošio novca na koju stavku iz ponude(unite sun eke vrednosti primera radi za cene), i petlja pretražuje ceo niz i izbacuje korisniku usluge i tim redosledom, respektivno, troškove za te stavke, ka oi na kraju ukupni trošak dodatnih usluga. Ovim smo implementirali funkciju petlji.

## Integraciono testiranje

Integraciono testiranja predstavlja metod testiranja koji sleduje posle jediničkog testiranja sistema. U okviru integracionog testiranja postoje dve vrste pristupa ovom testiranju : top-down i bottom-up tehnike i primenjuju se u zavisnosti od veličine programa kao i preferenci testera u nekim slučajevima. Integraciono testiranje se obavlja u toku razvoja softvera a ne na kraju istog. Ovo poboljšava otkrivanje grešaka i eventualnih defekata posmatranog sistema i poboljšava celokupan kvalitet softvera.

U primeru ovog projekta, koristićemo top-down tehinku integracionog testiranja:

Rezervacija

# Izveštaj o testiranju

**Datum:** 24.12.2022.

**Br. Izveštaja:** IZV-01

**Testeri:** Bojana Stajić

**Komponenta testiranja:** Pregled ponude hotela

**Testirano na:** Brave (Chromium) version 1.46.144

**Naziv defekta:** Loš prikaz hotela prilikom izlistavanja ponude istih

**Ozbiljnost:** Visoki prioritet

**Opis defekta:** Kada se učita ponuda hotela, izlistane su i zauzete sobe kao i one koje nisu u ponudi više.

**Uzroci i način otklanjanja:** Uvesti redovno automatsko ažuriranje podataka iz baze podataka sa listom ponuda hotelskih soba.

**Izveštaj o defektima**

**Izveštaj o defektima**

**Datum:** 24.12.2022.

**Br. Izveštaja:** IZV-02

**Testeri:** Bojana Stajić

**Komponenta testiranja:** Unos broja hotelskih soba

**Testirano na:** Brave (Chromium) version 1.46.144

**Naziv defekta:** Provera negativnih vrednosti

**Ozbiljnost:** Visoki prioritet

**Opis defekta:** Sistem dozvoljava unos negativnih brojeva kao brojeva soba za željenu rezervaciju.

**Uzroci i način otklanjanja:** Zabraniti čuvanje negativnih vrednosti ograničavanjem vrednosti koja može da se unese kao broj željene sobe.

# ODRŽAVANJE, DALJI RAZVOJ I KONTINUALNI PROCES POBOLJŠANJA SOFTVERA

## Model estimacije troškova prilikom održavanja odabrane aplikacije

***Vođa projekta*** - Radiće na upravljanju svih aktivnosti na projektu, dobijaće izveštaje od strane softverskih inženjera i administratora sistema.

Predviđeno radno vreme na projektu je 80 radnih sati. Cena rada po satu iznosi 35 evra. 80 \* 35 = 2800 evra

***Administrator sistema*** – Podešavanje softvera i hardvera u serverskom delu sistema, unos novih korisnika, lokacija, smeštaja.

Predviđeno radno vreme na projektu je 200 radnih sati. Cena rada po satu iznosi 20 evra. 200 \* 20 = 4000 evra

***Softverski inženjer*** ***1***– Biće zadužen za grafički korisnički interfejs sistema, za kreiranju dokumentacije i na planiranju modula za integraciju za dostavu klijentima.

Predviđeno radno vreme na projektu je 350 radnih sati. Cena rada po satu iznosi 50 evra. 350 \* 50 = 17500 evra

***Softverski inženjer 2*** – Biće zadužen za vezu starog sistema i novog koji se razvija, kako bi prelazak i integracija između ta dva bili još bolji, i na kreiranju modela baze podataka.

Predviđeno radno vreme na projektu je 350 radnih sati. Cena rada po satu iznosi 47 evra. 350 \* 47 = 16450 evra

***Tester*** – Angažovanje tokom testiranja sistema, njegovih osnovih funkcija kao i novih modula. Testiraće i bazu podataka sa velikim brojem unetih hotel ai njen rad kao i grafički interfejs sistema.

Predviđeno radno vreme na projektu je 40 radnih sati. Cena rada po satu iznosi 15 evra. 40 \* 15 = 600 evra

Ostali resursi:

• PC-Server – 18000 evra

• PC 1 – 900 evra

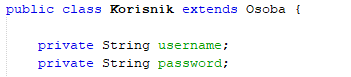
• PC 2 – 900 evra

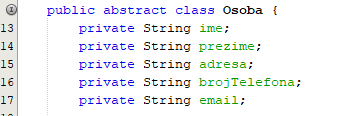
• PC 3 – 900 evra

• PC 4 – 900 evra

## Pristup za postizanje višekratne upotrebljivosti

Primer reusability klasa primenom nasleđivanja:





Generalizacija predstavlja tehinku koja na jednostavan način rešava kompleksne situacije. Podrazumeva principe nasleđivanja : uopštene klase sadrže atribute i operacije koje imaju druge klase, koje su u tom slulčaju podklase te uopštene klase. Podklase nasleđuju i atribute i metode svojih roditeljskih klasa pojednostavljujući kod sistema, opis klasa sistem, i čineći kod preglednijim i kvalitetnijim, izbegavajući da više klasa ima iste atribute i metode.

Primer primene design pattern-a

Šablon dekorator omogućava dodavanje nove funkcionalnosti nekom postojećem objektu, ali bez menjanja njegove strukture. Ovaj šablon kreira Decorator klasu koja je omotač te orginalne klase i obezbeđuje dodavanje tih dodatnih funkcionalnosti bez promena potpisa metoda inicijalne klase.

# ZAKLJUČAK

Zadovoljan korisnik predstavlja temelj svakog dobrog sistema, poslovanja ili celokupne franšize na jednom tržištu. Zbog toga je bitno obezbediti korisniku proizvod koji je visoko funkcionalan i optimalan koji podrazumeva pouzdanost. Softver treba da prati korisnički zahteve i potrebe praveći jedan sistem koji će odgovarati profile prosečnog korisnika sa težnjom da ispuni i prohteve onih malo manje konvencionalnijih zahteva. Testiranje softvera stavlja pečat na jedan uspešan softver dajući mu poslednji manifest njegove ispravnosti i ispunjenja svih korisničkih zahteva.

# LITERATURA

1. SE321 Obezbeđenje kvaliteta, testiranje i održavanje softvera, predavanja u elektronskom obliku objavljena na e-Learning sistemu. Univerzitet Metropolitan, 2022.