Vrsta zadatka: Projektna specifikacija

Projekat predstavlja pojednostavljenu i problematičnu implementaciju online prodavnice. U projektu se nalazi model, sloj za komunikaciju sa bazom podataka i sloj servisa, u kojem se nalazi trenutno implementirana poslovna logika.

1. Entiteti u sistemu su:
   - **Client** koji predstavljaju kupce, odnosno korisnike sistema. Imaju ime, korisničko ime i lozinku. Korisničko ime je jedinstveno u sistemu. Lozinka se čuva u bazi podataka u plain text formatu, odnosno nije enkriptovana. Ime, korisničko ime i lozinka su obavezni podaci, ne mogu da budu prazni.
   - **ShopItem**, što predstavlja proizvod u prodavnici. Svaki proizvod ima ime, cenu i količinu koja je dostupna na lageru. Cena i količina nikada ne mogu biti negativne vrednosti. Ime je takođe obavezno i ne može da bude prazno.
   - **DeliveryService**, što predstavlja dostavu. Korisnik može pri kupovini (na frontend delu aplikacije) da odabere dostavljača. Dostavljač ima naziv, početnu fiksnu cenu dostave, kao i dodatnu cenu dostave po kilometru. Podaci su obavezni i moraju da budu veći od 0.
   - **Transaction** predstavlja jednu transakciju, odnosno račun koji se formira kada korisnik kupi neki proizvod. Sadrži podatke o samom proizvodu, klijentu i odabranom dostavljaču (u vidu njihovog jedinstvenog identifikatora). Takođe ima podatak o odabranoj količini, koja ne može da bude veća od dostupne količine proizvoda, datumu formiranja transakcije, udaljenosti izraženoj u stotinama kilometara (1 = 100km) kao i ukupne cene, koja se računa pri formiranju transakcije. Udaljenost ne može da bude negativna.
   - **Promotions** - promocije. Usled obimnosti zadatka i nedostatka vremena, pre svega zbog grešaka pri dizajniranju i realizaciji rešenja, promocije nisu trenutno implementirane.
2. Sloj za komunikaciju sa bazom podataka podržava operacije dobavljanja svih torki tabele, dobavljanje jedne torke na osnovu id-a, ubacivanje, ažuriranje i brisanje jedne torke. Sve ostale operacije (login, pretraga...) se izvršavaju na serveru uz pomoć postojećih operacija.
3. Funkcionalnosti vezane za korisnika obuhvataju:
   - **login**, odnosno prijavu na sistem. Metoda prima korisničko ime i lozinku korisnika, dobavlja iz baze sve korisnike i proveravaju da li jedan korisnik ima isto korisničko ime i lozinku. Kada se pronađe korisnik čiji se podaci podudaraju sa prosleđenim, on se vrati iz metode. Ukoliko se ne pronađe nijedan korisnik, vraća se null. Usled ograničenja jedinstvenosti korisničkog imena, moguće je samo da postoji 1 ili 0 korisnika sa traženim podacima.
   - **register**, registracija na sistem. Prosleđuju joj se podaci za kreiranje korisnika, ime, korisničko ime i lozinka. Ukoliko se ne pronađe u bazi korisnik koji već ima to korisničko ime, kreira se novi korisnik. Ako je neko od polja prazno, registracija je neuspešna. Metoda vraća boolean podatak koji ukazuje na to da li je registracija uspešna.
   - **deleteUser**, brisanje korisnika, ukoliko korisnik postoji.
   - **updateInfo**, ažuriranje podataka korisnika. Prima celog korisnika, novo ime, novu lozinku i staru lozinku. Ukoliko postoji korisnik, odnosno login uspe, korisniku se podešava novo ime i nova lozinka, pa se čuva u bazi.

4. Funkcionalnosti vezane za proizvod su:

   - **postItem**, za kreiranje novog proizvoda u prodavnici. Prima ime, cenu i količinu. Proverava validnost podataka, i zatim ubacuje novi proizvod u bazu.
   - **removeItem**, brisanje proizvoda iz prodavnice. Proverava se da li proizvod ima id, pa se briše iz baze ukoliko ima.
   - **buy**, za kupovinu odnosno umanjenje količine proizvoda na lageru. Prima proizvod koji se kupuje i kupljenu količinu. Proverava se da li je prosleđen negativan broj za količinu, pa se čuva u bazi ukoliko nije.
   - **stockUp**, za uvećanje količine proizvoda na lageru. Kao i metoda za umanjenje količine, prima proizvod i količinu. Ukoliko količina nije negativna, na postojeću količinu proizvoda se doda nova, i sačuva se u bazi.
   - **checkIfPopular**, koja proverava da li je proizvod trenutno popularan. Prima kao argument proizvod koji se proverava, a vraća podatak o tome da li je proizvod popularan ili ne. Za proveru popularnosti se pregledaju sve transakcije izvedene u prethodnih 30 dana. Proizvod je popularan ako:
     - Je njegova cena veća od 300 i količina kupljena u prethodnih 30 dana je veća od 60% trenutnih zaliha proizvoda, ili
     - njegova cena je manja od 300, i količina kupljena u prethodnih 30 dana je veća od 80% trenutnih zaliha proizvoda.
   - **getTrendingIndex**, za dobavljanje indeksa traženosti proizvoda, odnosno koliko je zapravo tražen proizvod. Ukoliko proizvod nema id, odnosno nije validan proizvoda, vraća IllegalArgumentException. Za računanje indeksa traženosti, posmatraju se sve transakcije i prikuplja ukupan profit za sve transakcije sa ovim proizvodom. Taj profit se zatim deli sa brojem dana od poslednje transakcije u kojoj je kupljen proizvod, odnosno formula je:

$$indeksTrazenosti = ukupanProfit/brojDanaOdPoslednjeKupovine \tag{1}$$

5. Funkcionalnosti vezane za dostavu su:

   - **register**, registracija novog dostavljača. Prima podatke koji su potrebni za kreiranje dostavljača, ime, cenu i cenu po kilometru. Ukoliko su podaci validni, kreira se novi dostavljač i ubacuje u bazu podataka.
   - **deleteDeliveryService**, brisanje dostavljača iz baze podataka.
   - **updateInfo**, ažuriranje podataka dostavljača. Prima objekat koji se ažurira, novo ime, novu početnu cenu i novu cenu po kilometru. Ukoliko su podaci validni, postavljaju se postojećem objektu i objekat se ažurira u bazi podataka.

6. Funkcionalnosti vezane za transakcije su:

   - **completeTransaction**, za sklapanje, odnosno formiranje transakcije. Prima korisnika, dostavljača, proizvod, kupljenu količinu i udaljenost do adrese za dostavu, izražene u stotinama kilometara. Proverava se da li su podaci validni, pa, ako jesu, formira se nova transakcija, računa se konačna cena, poziva se kupovina proizvoda (umanjenje količine na lageru) i transakcija se čuva u bazi podataka. Ako podaci nisu validni, prouzrokuje IllegalArgumentException.
   - **calculatePrice**, za računanje konačne cene kupovine. Prima proizvod, udaljenost, količinu i dostavljača. Za početak se saberu početna cena dostave i cena proizvoda, pri čemu, ako je kupljeno više od 20, a manje od 50 proizvoda, cena proizvoda se smanjuje za 10%. Ako je kupljeno 50 ili više proizvoda, cena se smanjuje za 20%. Takođe se pri računanju cene uzima u obzir popust na bliske dostave, odnosno cena se umanjuje za bliske dostave prema formuli:

$$cena- = 350/udaljenost * 20 \tag{2}$$

Udaljenost se ovde zaokružuje na ceo broj, zbog jednostavnijeg računanja. Zatim se na cenu dodaje nadoknada za udaljenost, odnosno $udaljenost * 100 * cenaPoKilometru$ (zbog udaljenosti izražene u stotinama kilometara). Na konačno dobijenu cenu se zatim primenjuju promocije, ukoliko postoje. Metode za primenu i proveru promocija su trenutno samo placeholder-i i vraćaju uvek iste vrednosti, ali postupak je sledeći:

- uračunaju se specijalne promocije, pozivanjem metode `calculateSpecialPromotions`,
- proverava se da li je trenutno neki praznik, pozivanjem metode `checkHolidays`, koja prima trenutni datum,
- Ukoliko jeste neki praznik, uračunaju se sezonske promocije, pozivanjem metode `calculateSeasonalPrommotion` sa trenutnim datumom,
- računa se popust, pozivanjem metode `calculateDiscount`
- vraća se popust.

Bitno je da se tačno izvršava navedenim redosledom, i da se nijedna metoda ne izvršava više puta nego što je navedeno. Kada se sračuna popust, proverava se da li je popust veći od pola cene. Ako jeste, ne primenjuje se, a ako nije, primeni se. Na kraju se vraća cena transakcije.

- **getRecentTransactions** za dobijanje transakcije izvršenih u poslednjih 30 dana. Dobavlja sve transakcije, i vraća samo one čiji je datum izvršavanja veći od $trenutniDatum - 30dana$.