Laboratorijska vježba br. 1

Izvještaj o Inspekciji Koda

Check liste inspekcije koda

Označiti stavke na listama za inspekciju koje su ispunjene, nakon vršenja inspekcije koda. Za stavke koje se ne označe potrebno je navesti detaljne informacije o greškama u nastavku.

Inspekcija strukture programskog rješenja

Ova lista stavlja fokus na potencijalne probleme u strukturi programskog koda, analizi koda na visokom nivou i poštovanju standarda.

☒ Kod je napisan u skladu sa važećim standardima kodiranja.
☒ Stil kodiranja je konzistentan u cijelom programskom rješenju.
☒ Kod je ispravno formatiran.
☒ U kodu nema funkcija koje se ne pozivaju ni na jednom mjestu.
☒ Nema nedostižnih linija koda.
☒ Nema bespotrebnog implementiranja funkcija koje mogu biti zamijenjene postojećim bibliotekama.
☒ U kodu nema ponavljanja koje može biti zamijenjeno jedinstvenom funkcijom.
☐ Memorija se koristi na efikasan način.
☒ Nema korištenja *magičnih brojeva* i konstanti bez korištenja varijabli.
☐ Nema previše dugih i kompleksnih blokova koda.

Inspekcija dizajna programskog rješenja

Ova lista stavlja fokus na potencijalne probleme u poštovanju objektno-orijentisanih principa, SOLID principa i dizajn pattern-a u okviru programskog rješenja.

☒ Svaka klasa ima malu kompleksnost i jedan tip operacija i zaduženja.
☒ Klase su prilagodljive budućim promjenama.
☒ Svi objekti izvedenih klasa zamjenjivi su svojim osnovnim klasama.
☐ Interfejsi su jednostavni, s malim brojem funkcija.
☒ Dubina nasljeđivanja nije velika.
☒ Klijent može jednostavno pristupati objektima kontejnerskih klasa, bez potrebe definisanja detalja gradivnih dijelova klase.
☒ U slučaju potrebe ponovnog korištenja većeg broja istih objekata, objekti se ne instanciraju više puta.
☒ Instanciranje kontejnerske klase vrši se samo jednom.
☐ Sigurnost aplikacije osigurana je putem *proxy*-a.

Inspekcija varijabli i izraza programskog rješenja

Ova lista stavlja fokus na potencijalne probleme u strukturi koda na visokom nivou, uključujući varijable i izraze u kodu.

☒ Sve varijable imaju imena koja odgovaraju njihovoj namjeni.
☒ Koristi se jedan stil imenovanja varijabli.
☒ Nema varijabli koje se ne koriste.
☒ Nema neosiguranih potencijalnih dijeljenja s nulom.
☐ Operator = ne koristi se u logičkim izrazima.

Inspekcija petlji i grananja programskog rješenja

Ova lista stavlja fokus na potencijalne probleme u petljama i grananjima u kodu.

☒ Nema praznih niti nedostižnih blokova koda.
☒ U *if* blokovima testiraju se češći scenariji.
☒ Svi *switch* iskazi imaju definisan *default* slučaj.
☒ Sve petlje imaju uslov završetka.
☒ Nema velikog broja gniježdenja petlji.
☒ U petljama nema koda koji se može izvršiti izvan petlje.

Inspekcija memorijskih operacija programskog rješenja

Ova lista stavlja fokus na potencijalne probleme u korištenju memorije te konekciji s bazama podataka, vanjskim uređajima i korištenjem file-ova u kodu.

☒ Sve varijable koje koriste indeksiranje su inicijalizirane prije korištenja.
☒ Sva alocirana memorija dealocira se prije završetka izvršavanja.
☒ Pri radu s vanjskim uređajima, postoji provjera za *timeou0t*.
☒ Prije pokušaja modificiranja *file*-ova, provjerava se da li oni postoje.
☐ Nakon završetka transakcije, konekcija s bazom podataka se uvijek zatvara.

Inspekcija dokumentacije programskog rješenja

Ova lista stavlja fokus na potencijalne probleme u razumljivosti i jednostavnosti dokumentovanja koda.

☐ Svi kompleksni dijelovi koda posjeduju komentare.
☒ Dijelovi koda podijeljeni su u regije.
☐ Metode klasa imaju svoje opise.
☒ U cijelom rješenju koristi se jedan stil komentarisanja koda.

Informacije o timu koji vrši inspekciju koda

Popuniti informacije o članovima tima koji vrši inspekciju.

Ime i prezime, broj indexa: Kerim Gazić, 226-ST
Zaduženje: Sistem kreiranja knjiga
Predmet inspekcije: Kod modula za ovaj sistem

Ime i prezime, broj indexa: Click here to enter text.
Zaduženje: Click here to enter text.
Predmet inspekcije: Click here to enter text.

Izvještaj o pronađenim greškama

Popuniti informacije o pronađenim greškama, te kategorijama u koje spadaju. Lokacija greške u modulu podrazumijeva file i linije koda u kojima se greška nalazi.

Br.	*Check Lista*	*Tip*	*Opis*	*Lokacija*	*Ozbiljnost*
1.	Defensive Programming – validation of user input	High severity bug	Program koristi `int.Parse()` bez TryParse, što izaziva pad programa na nevalidan unos.	Program.cs, više linija (npr. linija 52, 96, 117)	High
2.	Defensive Programming – null handling	High severity bug	`Search(null)` uzrokuje `NullReferenceException` u `query.ToLower()`.	LibraryInventory.cs, linija 32	High
3.	Documentation	Medium	Kod nije dokumentovan; nema komentara ni objašnjenja kompleksnih dijelova.	cijeli projekat	Medium
4.	Loops and Branches – updating variables	Medium	Ažuriranje korisnika dozvoljava unos praznih polja bez validacije.	Program.cs, linija ~63 i dalje	Medium
5.	Structure – duplicated code	Minor	Parsiranje ID-a ponavlja se u više funkcija. Može se refaktorisati u metodu.	Program.cs	Minor
6.	Variables – unused/undefined	Minor	Ne postoji posebna provjera indeksa prije pristupa listi knjiga.	LibraryInventory.cs (UpdateBook, DeleteBook)	Minor

7.	Defensive Programming – exception type	Medium	Baca se generički `Exception` umjesto preciznog tipa (`InvalidOperationException`).	UserService.cs, linija 32	Medium
8.	Structure – efficiency	Minor	`AddBook()` ne provjerava duplikate knjiga, pa lista može nekontrolisano rasti	LibraryInventory.cs, linija 10	Minor
9.	Documentation – comments consistency	Minor	Nedostatak bilo kakvih komentara otežava razumijevanje koda.	cijeli projekat	Minor
10.	Structure – nesting & design	Minor	Dugi switch-case blokovi mogli bi se zamijeniti funkcionalnim handler metodama.	Program.cs (meni logika)	Minor

Izvještaj o metrikama grešaka

Ukupan broj pronađenih grešaka: 10

Normirani broj grešaka: 390

Broj grešaka po LOC:0.0256

Broj normiranih grešaka po LOC: 2.56 greške / 100 LOC

Efikasnost otkrivanja grešaka: **Efikasnost otkrivanja: 0.75 (75%)**

Normirana efikasnost otkrivanja grešaka: 1.92