

## **Zadaća br. 1** **Izveštaj o inspekciji koda**

### ***Uputstvo za izradu zadaće***

*Izrada zadaće vrši se u formi izvještaja koja je data u nastavku. Potrebno je popuniti sva polja data u izvještaju, odgovoriti na pitanja i dodati tražene slike. Nije dozvoljeno brisati postojeća, niti dodavati nova polja.*

*Zadaća se radi u timovima od po tri studenta. Svi studenti iz istog tima popunjavaju isti izvještaj u jednom dokumentu, s tim da popunjavaju različite dijelove dokumenta ovisno o postavkama zadataka. Dovoljno je da jedan član tima pošalje izvještaj preko Zamgera.*

### ***Informacije o timu***

*Popuniti informacije o studentima koji vrše izradu zadaće.*

Dodijeljeno programsko rješenje: StudentskiDom

Ime i prezime: Lejla Pirija  
Broj indexa: 18238

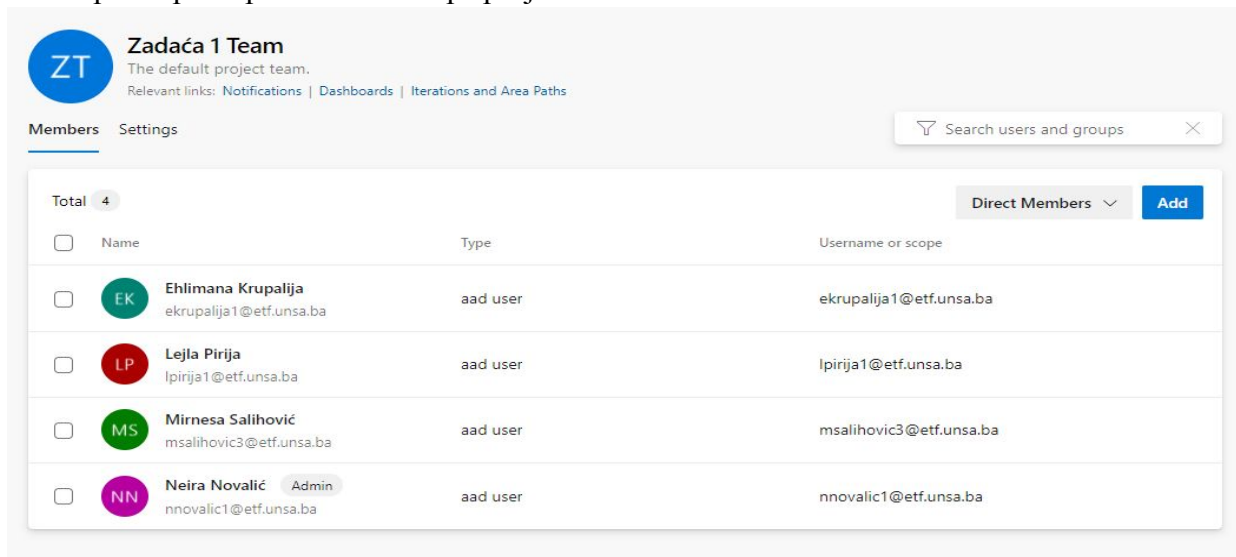
Ime i prezime: Neira Novalić  
Broj indexa: 18112

Ime i prezime: Mirnesa Salihović  
Broj indexa: 18115

## Zadatak 1. (Konfiguracija okruženja)

Potrebno je izvršiti konfiguraciju Azure DevOps organizacije te kreirati projekat kojem će imati pristup svi članovi tima, kao i predmetni asistent nastavne grupe.

Prikaz prava pristupa Azure DevOps projektu:



| <input type="checkbox"/> | Name  | Type     | Username or scope       |
|--------------------------|---|----------|-------------------------|
| <input type="checkbox"/> | <b>EK</b> Ehlimana Krupalija<br>ekrupalija1@etf.unsa.ba             | aad user | ekrupalija1@etf.unsa.ba |
| <input type="checkbox"/> | <b>LP</b> Lejla Pirija<br>lpirija1@etf.unsa.ba                      | aad user | lpirija1@etf.unsa.ba    |
| <input type="checkbox"/> | <b>MS</b> Mirnesa Salihović<br>msalihovic3@etf.unsa.ba              | aad user | msalihovic3@etf.unsa.ba |
| <input type="checkbox"/> | <b>NN</b> Neira Novalić <span>Admin</span><br>nnovalic1@etf.unsa.ba | aad user | nnovalic1@etf.unsa.ba   |

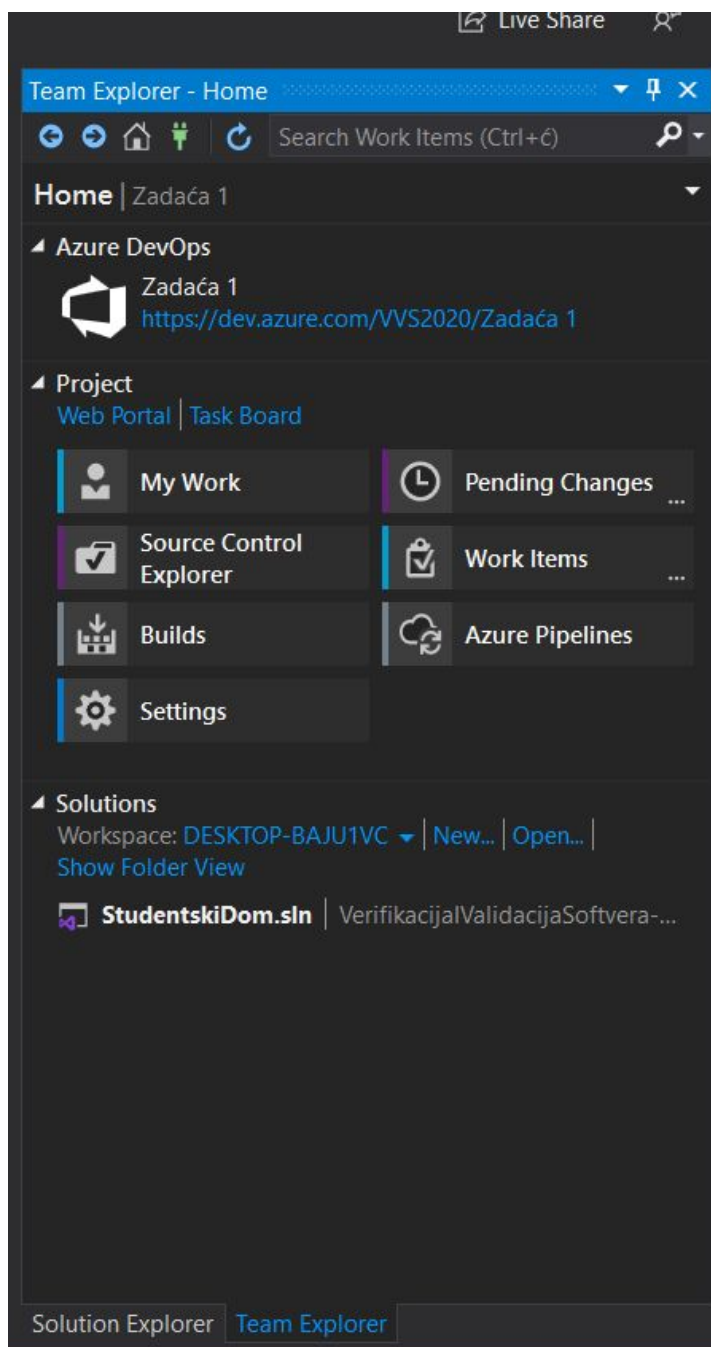
Da li bi bilo moguće commitati programsko rješenje na GitHub repozitorij, a zatim GitHub repozitorij povezati sa DevOps projektom? Ukoliko ne, zašto? Ukoliko da, da li će se taj pristup koristiti pri izradi ove zadaće?

Moguće je commitati programsko rješenje na GitHub repozitorij, a zatim GitHub repozitorij povezati sa DevOps projektom. Pri izradi ove zadaće neće se koristiti takav pristup, jer zadaća uključuje inspekciju koda, a kada bismo povezali GitHub repozitorij sa DevOps projektom, ne bi imali mogućnost inspekcije koda, jer bi se on tada prepoznao kao i ostali git repozitoriji.

Potrebno je povezati se sa DevOps projektom koristeći Visual Studio okruženje. Zatim je dodijeljeno programsko rješenje potrebno commitati na Azure DevOps koristeći TFVC.

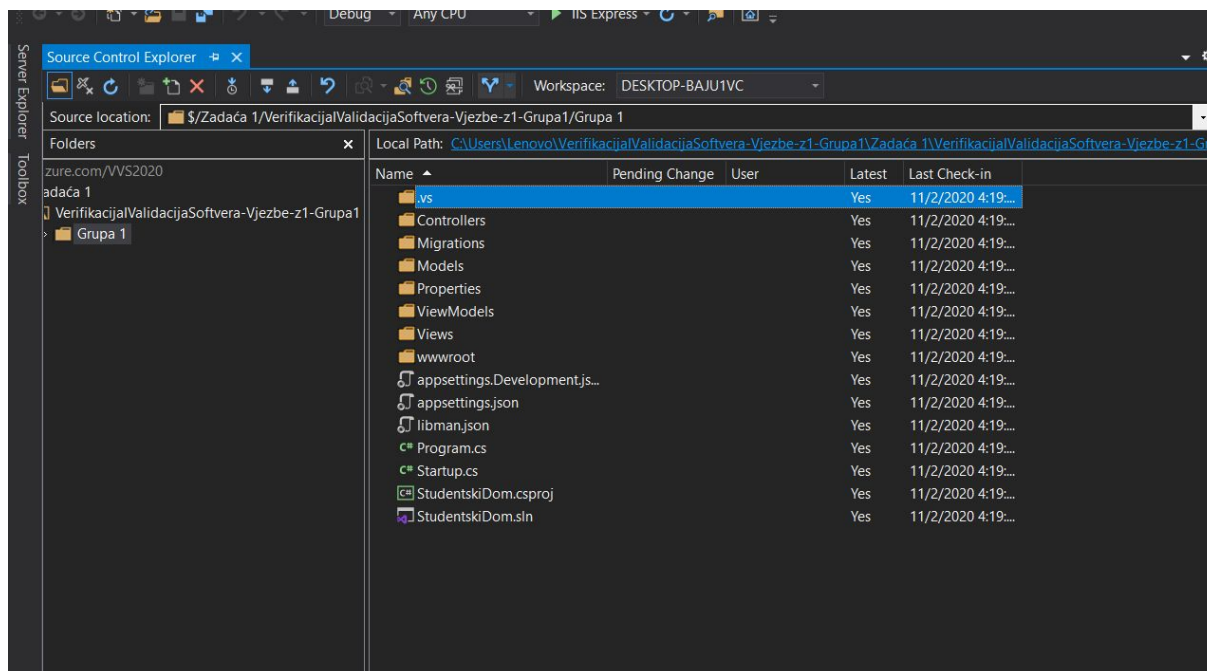
Prikaz Team Explorer taba u Visual Studio okruženju nakon konekcije na Azure DevOps server:

Verifikacija i Validacija Softvera

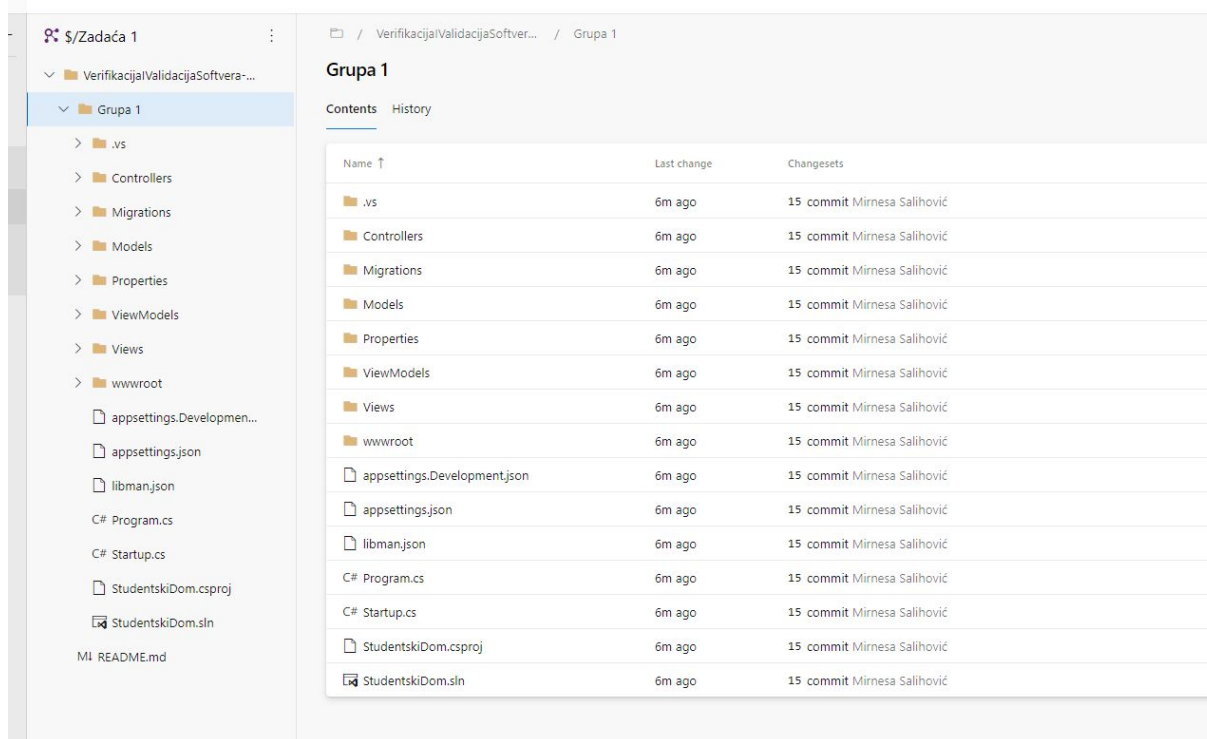


Prikaz *Source Control Explorer* taba u Visual Studio okruženju sa repozitorijem projekta:

## Verifikacija i Validacija Softvera



Prikaz *Repos* taba u Azure DevOps okruženju:



Kakva je razlika između dva prikaza sadržaja repozitorija iznad? Kada je pogodnije koristiti prikaz u Visual Studio, a kada prikaz u Azure DevOps okruženju?

*Verifikacija i Validacija Softvera*

U Visual Studio možemo pratiti promjene koje se trenutno vrše nad nekom datotekom, odnosno vidimo trenutno stanje projekta. A da bismo vidjeli promjene na Azure DevOps repozitoriju moramo uraditi Check in. Dakle, u Visual Studio, da bismo vidjeli posljednje promjene koje su napravljene, moramo uraditi desni klik na View History, dok u Azure DevOps-u odmah vidimo promjene koje su napravljene.

***Zadatak 2. (Walkthrough plan inspekcije koda)***

*Prije vršenja inspekcije koda, potrebno je napraviti plan vršenja inspekcije. Prvo je potrebno dodijeliti dijelove programskog rješenja članovima tima tako da svi članovi tima dobiju otprilike jednake dijelove programskog koda za inspekciju. Ispod je potrebno označiti koji dijelovi koda su dodijeljeni pojedinačnim članovima tima.*

Klase dodijeljene članu 1: Algoritam, Bagajna, ErrorViewModel, Paviljon, Pol, RedovanStudent, Restoran, Student, StudentskiDonContext, Uprava, ZahtjevRestorana, zahtjevZaNabavkiNamirnica, AdministratorController, Isoba, IStudent

Klase dodijeljene članu 2: AzurirajMeni, DnevniMeni, GodinaStudijaSort, Mjesec, PrebivalisteInfo, PregledStanjaBonova, RasporedKanton, StavkaNarudzbe, StudentPonovac, Večera, ZahtjevStudenta, ZahtjevZaPremjestaj, StudentController

Klase dodijeljene članu 3: AzurirajStanjeBonova, KantonFilter, Korisnik, LicniPodaci, Proxy, RasporedFakultet, SkolovanjeInfo, Soba, StudentskiDonSingleton, Zahtjev, ZahtjevZaCimeraj, ZahtjevZaUpis, UpravaController, IstudentskiDom, Ručak, HomeController

*Za dodijeljene klase trebaju se odrediti jednostavni testni slučajevi koji će se koristiti za mentalno izvršavanje kako bi se pri inspekciji lakše pronašle greške. U nastavku svi članovi tima trebaju dati po jedan primjer testnog slučaja za neke od kompleksnijih dijelova koda koji su im dodijeljeni.*

Prikaz programskog koda jednog od kompleksnijeg dijela koda (član 1):

```
public async Task<IActionResult> EditUsersInRole(string roleId)
{
    ViewBag.roleId = roleId;

    var role = await roleManager.FindByIdAsync(roleId);

    if (role == null)
    {
        ViewBag.ErrorMessage = $"Role with Id = {roleId} cannot be found";
        return View("NotFound");
    }

    var model = new List<UserRoleViewModel>();

    foreach (var user in userManager.Users.ToList())
    {
        var userRoleViewModel = new UserRoleViewModel
        {
            UserId = user.Id,
            UserName = user.UserName
        };

        if (await userManager.IsInRoleAsync(user, role.Name))
        {
            userRoleViewModel.IsSelected = true;
        }
        else
        {
            userRoleViewModel.IsSelected = false;
        }

        model.Add(userRoleViewModel);
    }

    return View(model);
}
```

Opis testnog slučaja za kod prikazan iznad:

Testni slučaj :Provjera ispravnosti EditUsersInRole metode u Administrator kontroleru

Testni slučaj: Izabrati rolu sa željenim id-em i provjeriti da li će u listi biti ispravno označeni korisnici kojima je rola dodijeljena i oni kojima rola nije dodijeljena.

Koja mjesta u kodu iznad imaju najveću vjerovatnoću greške? Koje vrste grešaka sa kojom ozbiljnošću mogu nastati na tim mjestima?

Najveću vjerovatnoću greške ima dio koda koji vrši manipulaciju sa korisnicima (prema tome da li im je rola dodijeljena ili ne). Ukoliko neki dio ovog kontrolera ne radi ispravno, sprječava se postizanje osnovnih mogućnosti i to je greška tipa 5.

Prikaz programskog koda jednog od kompleksnijeg dijela koda (član 2):

```
return View(student);  
}  
  
// GET: Student/Delete/5  
[HttpGet]  
public async Task{  
    if (id == null)  
    {  
        return NotFound();  
    }  
  
    var student = await _context.Student  
        .Include(s => s.LicniPodaci)  
        .Include(s => s.PrebivalisteInfo)  
        .Include(s => s.SkolovanjeInfo)  
        .Include(s => s.Soba)  
        .FirstOrDefaultAsync(m => m.Id == id / 0);  
    //kraj sumjernog dijela koda/  
  
    if (student == null)  
    {  
        return NotFound();  
    }  
  
    return View(student);  
}  
  
// POST: Student/Delete/5  
[HttpPost, ActionName("Delete")]  
[ValidateAntiForgeryToken]  
[ValidateAntiForgeryToken]  
public async Task{  
    var student = await _context.Student.FindAsync(id);  
    foreach (var s in _context.Student)  
    {  
        if (s.Id == id)  
        {  
            _context.Student.Remove(s);  
        }  
    }  
}
```

Opis testnog slučaja za kod prikazan iznad:

Testni slučaj :Provjera ispravnosti Delete metode u Student kontroleru

Testni slučaj: Izabrati studenta sa željenim id-em i provjeriti da li će se ukloniti iz liste

Koja mjesta u kodu iznad imaju najveću vjerovatnoću greške? Koje vrste grešaka sa kojom ozbiljnošću mogu nastati na tim mjestima?

Iako je veći dio kontrolera automatski generisan, postoji određen broj izmjena u kodu koje je potrebno provjeriti.

Najveću vjerovatnoću greške imaju mjesta gdje po id-u vršimo manipulacije sa listom registrovanih studenata.

Ukoliko sve funkcionalnosti u sklopu ovog kontrolera ne rade ispravno, ugrožene su osnovne funkcionalnosti aplikacije i to spada u opseg greške 3.



Prikaz programskog koda jednog od kompleksnijeg dijela koda (član 3):

```
public Soba RasporediStudenta(Student student)
{
    Soba slobonda = null;
    foreach (Paviljon p in StudentskiDomSingleton.Context.Paviljon.ToList())
    {
        p.Sobe = StudentskiDomSingleton.Context.Soba.Where(s => s.PaviljonId == p.PaviljonId).ToList();

        foreach(Soba s in p.Sobe)
        {
            s.Students = StudentskiDomSingleton.Context.Student.Where(st => st.SobaId == s.SobaId).ToList();

            if (s.DaLiImaMjesta())
            {
                if (slobonda == null && s.Students.Count == 0)
                {
                    slobonda = s;
                }
                else
                {
                    foreach (Student st in s.Students)
                    {
                        st.PrebivalisteInfo = StudentskiDomSingleton.Context.PrebivalisteInfo.FirstOrDefault( pi => pi.PrebivalisteInfoId == st.PrebivalisteInfoId);

                        if (st.PrebivalisteInfo.Kanton.Equals(student.PrebivalisteInfo.Kanton))
                        {
                            return s;
                        }
                    }
                }
            }
        }
    }

    return slobonda;
}
```

Opis testnog slučaja za kod prikazan iznad:

Testni slučaj: Testiranje klase RasporedKanton

Testni slučaj: Testiranje ispravnosti metode RasporediStudenta. Testiranje da li će studentu biti pravilno dodijeljena soba.

Koja mjesta u kodu iznad imaju najveću vjerovatnoću greške? Koje vrste grešaka sa kojom ozbiljnošću mogu nastati na tim mjestima?

Mjesto u kodu koja ima najveću mogućnost greske jeste pozivanja metode DaLiImaMjesta().

Ukoliko se desi greska u if uslovu metoda moze da vrati pogresno rjesenje sto moze izazvati greške u kodu tj. netačno dodjeljivanje sobe tj rasporeda studenata po sobama. Ono sto je bitno da priliko greske u ovoj metodi program će i dalje raditi međutim uzrokovat će greške u funkcionalnosti. Međutim rješenje kojim se greška tj. nejasnoće u funkcionalnosti može izbjeći je pozna pa ovaj dio koda bi mogao da bude vrsta greške 3. Također ovaj problem može biti i vrsta 5. Jer ugrozava sigurnost tačnosti izvršavanja projekta.

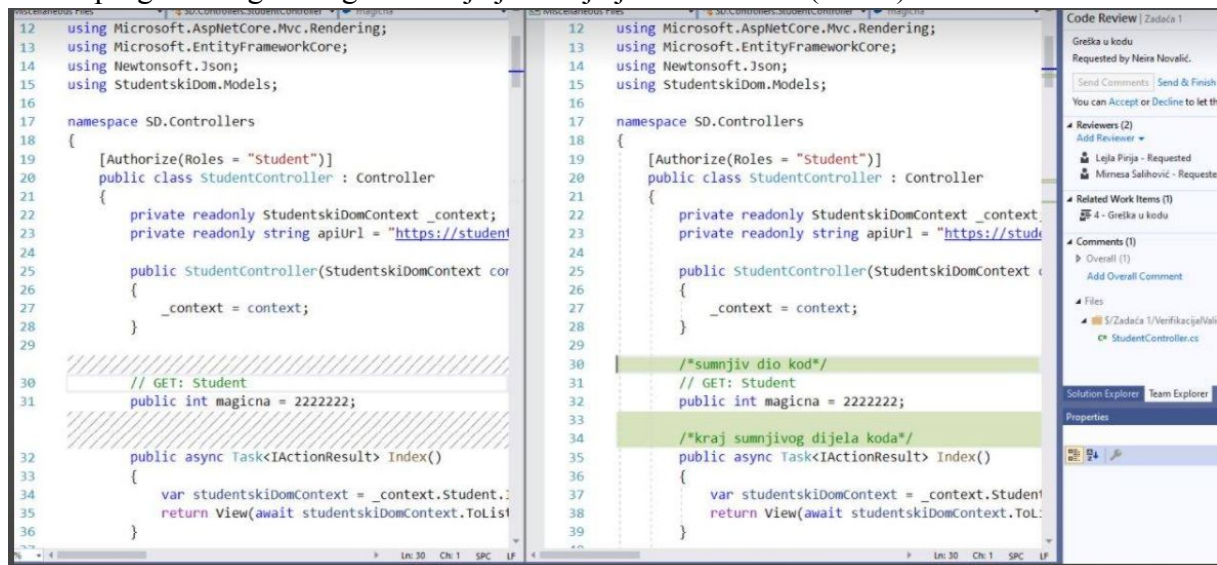


### Zadatak 3. (Inspekcija koda)

Svi članovi tima trebaju izvršiti inspekciju dodijeljenog programskog rješenja na način da različiti članovi tima pregledaju različite dijelove onako kako je to određeno u prethodnom zadatku. Svaku pronađenu grešku potrebno je dodijeliti ostalim članovima tima koristeći Code Review funkcionalnost Visual Studio okruženja. Svi članovi tima trebaju dobiti otprilike isti broj grešaka za code review.

U nastavku svaki član tima treba zabilježiti prvu grešku koja im je dodijeljena za inspekciju koristeći TFVC.

Prikaz programskog koda greške koja je dodijeljena za review (član 1):

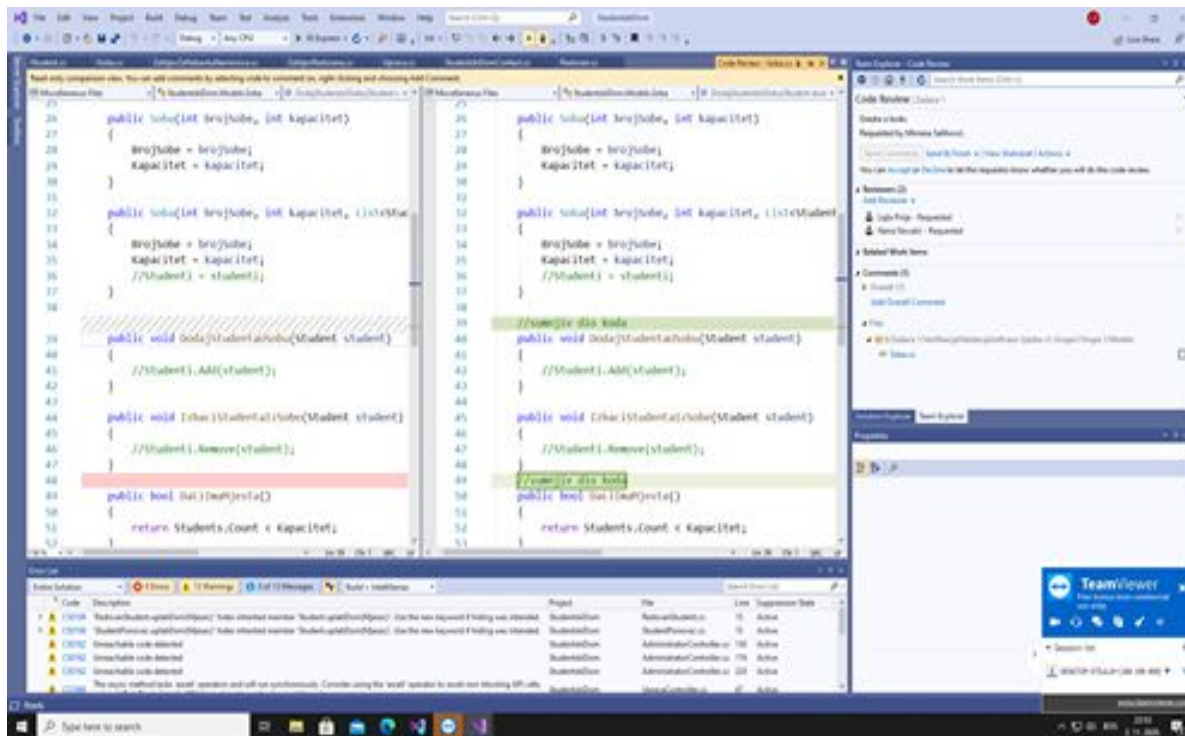


Check-lista kojoj gore prikazana greška pripada: Inspekcija strukture

Inspekcije strukture programskog rjesenja (koristenje magicnih brojeva)

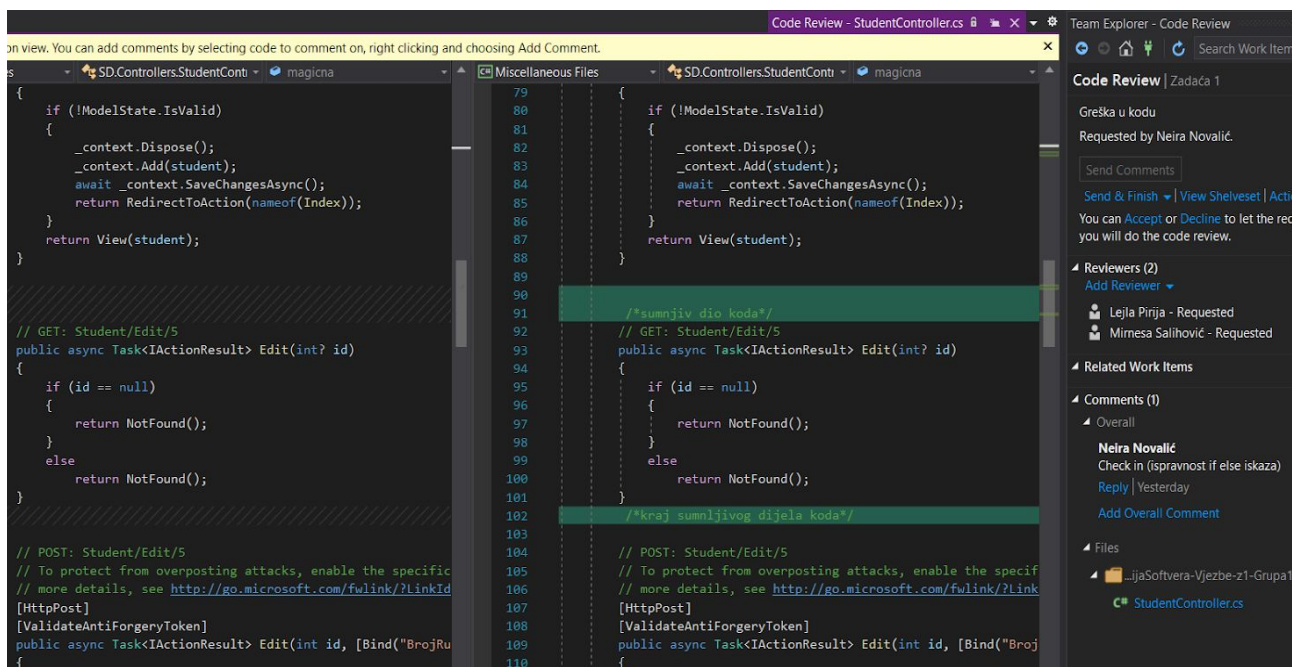
## Verifikacija i Validacija Softvera

Prikaz programskog koda greške koja je dodijeljena za *review* (član 2):



Check-lista kojoj gore prikazana greška pripada: Inspekcija strukture  
(Funkcija koja se ne poziva ni na jednom mjestu ; Neimplementirane metode sa zakomentarisanim kodom)

Prikaz programskog koda greške koja je dodijeljena za *review* (član 3):



Verifikacija i Validacija Softvera

Check-lista kojoj gore prikazana greška pripada: Inspekcija petlji i grananja

Da li sve gore prikazane greške pripadaju istoj check-listi? Da li postoji veća vjerovatnoća da greške pripadaju istoj check-listi i zašto?

Greške ne pripadaju istoj check-listi. Smatramo da postoji veća vjerovatnoća da greške pripadaju istoj check-listi zato što način programiranja i principi koje slijedi jedan programer najčešće dovode do istog tipa grešaka. To se potencijalno može odraziti ili na kvalitet koda ili na sam dizajn programskog rješenja.

Potrebno je identificirati sve pronađene greške te ih zabilježiti u tabelu koja se nalazi u nastavku. Svaki član tima treba pronaći minimalno po 3 greške.

| Br. | Check Lista                   | Opis greške   | Lokacija u kodu  | Ozbiljnost |
|-----|-------------------------------|---|--|------------|
| 1.  | Inspekcija varijabli i izraza | Sve varijable nemaju imena koja odgovaraju njihovoj namjeni | Klasa RasporedKanton metoda RasporediStudenta  | 3          |
| 2.  | Inspekcija strukture          | metode nisu iskoristene nigdje u programu                   | Klasa Soba, metode DodajStudentaUSobu iIzbaciStudentaIzSobe, ZahtjevZaSUpis konstruktor sa parametrima | 2          |
| 3.  | Inspekcija petlji i grananja  | Veliki broj ugnježenih petlji                               | Klasa RasporedKanton, metoda RasporediStudenta   | 2          |
| 4.  | Inspekcija petlji i grananja  | Nedostižni dijelovi koda,                                   | Klasa HomeController, metoda Login   | 2          |
| 5.  | Inspekcija strukture          | Korištenje magične konstante                                | Klasa Blagajna, metoda AzurirajStanjeVeceraAsync   | 1          |
| 6.  | Inspekcija varijabli i izraza | Dijeljenje s nulom  | Klasa Blagajna, metoda UplatiDomZaOdabraniMjesec   | 3          |
| 7.  | Inspekcija strukture          | Nedostižni dio koda   | Klasa AdministratorController, metoda EditRole   | 2          |
| 8.  | Inspekcija varijabli i izraza | Dijeljenje s nulom  | Klasa AdministratorController, metoda EditUsersInRole  | 3          |
| 9.  | Inspekcija petlji i grananja  | Petlja nema uslov završetka                                 | Klasa AdministratorController, metoda EditUsersInRole  | 3          |
| 10. | Inspekcija strukture          | Korištenje magične konstante                                | StudentController  | 1          |
| 11. | Inspekcija petlji i grananja  | If -else iskaz ima dvije iste povratne vrijednosti          | StudentController, metoda Details i Edit   | 3          |
| 12. | Inspekcija varijabli i izraza | Dijeljenje s nulom  | StudentController, metoda Delete   | 3          |

Verifikacija i Validacija Softvera

|     |                          |  |                   |   |
|-----|--------------------------|--|-------------------|---|
| 13. | Inspekcija strukture     | Magična konstanta u kodu                                     | GodinaStudijaSort | 1 |
| 14. | Inspekcija strukture     | Neimplementirane metode u klasi;Zakomentaran veliki dio koda | ZahtjevStudenta   | 1 |
| 15. | Inspekcija dokumentacije | Kompleksni dijelovi koda ne posjeduju komentare              | StudentController | 1 |

Da li su pronađene greške na mjestima koja su označena kao vjerovatna za pojavu grešaka u zadatku 2? Šta to govori o *walkthrough* planu inspekcije koda?

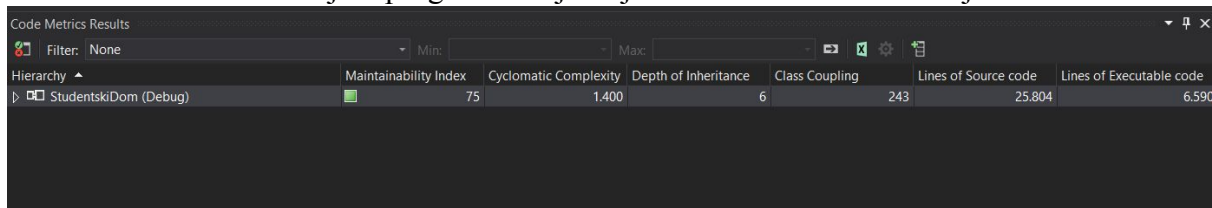
Da greške koje su pronađene obuhvataju greške koje su navedene u zadatku 2. Walkthrough plan inspekcije koda je efektan način pregleda koda i pronalaska grešaka. Analiza izvršavanja dijelova programskog koda „u našoj glavi“ nam daje bolji uvid o tome kako kod radi i familijariziranjem sa kodom laže možemo pronaći greške koje se u njemu nalaze.

#### Zadatak 4. (Metrike detekcije grešaka)

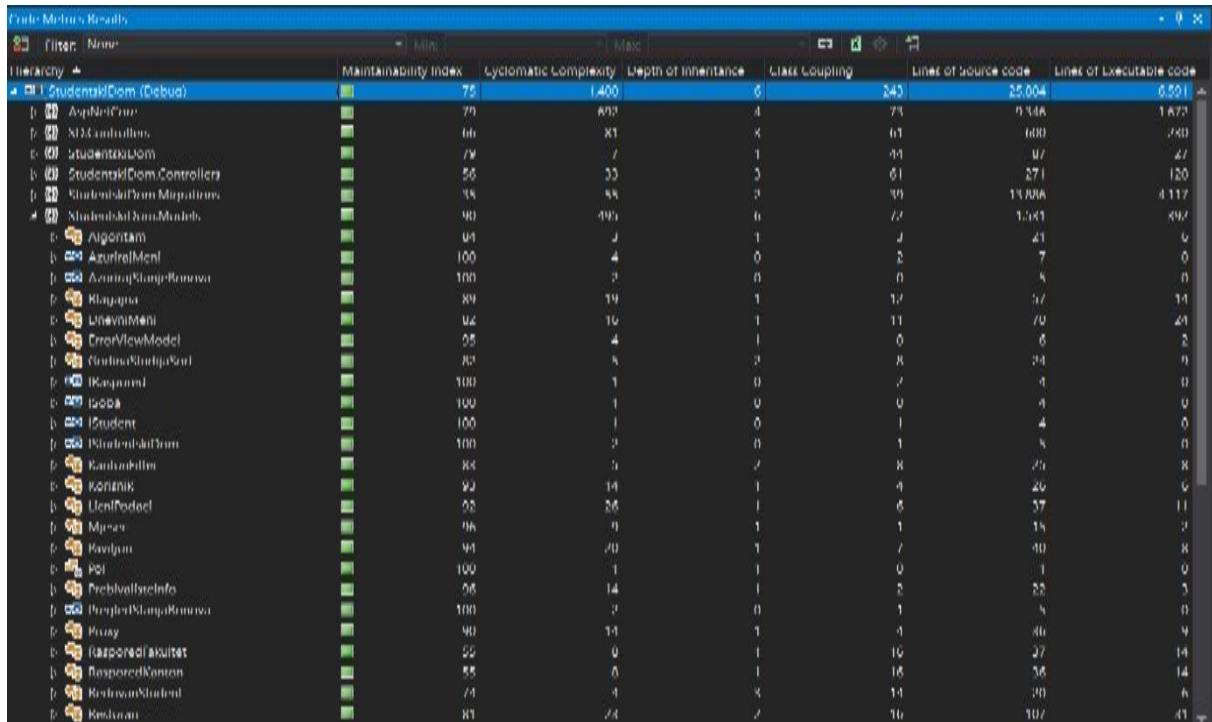
Za izračun metrika detekcije grešaka, prvo je potrebno napraviti sumarni izvještaj o pronađenim greškama. Sumarni izvještaj unosi se u tabelu koja se nalazi u nastavku. Za dobivanje normiranog broja grešaka potrebno je pomnožiti broj grešaka sa  $2^{\text{faktor ozbiljnosti greške}}$ .

| Ozbiljnost    | Broj grešaka | Normirani broj grešaka |
|---------------|--------------|------------------------|
| 1             | 5            | 10                     |
| 2             | 4            | 16                     |
| 3             | 6            | 48                     |
| 4             | 0            | 0                      |
| 5             | 0            | 0                      |
| <b>Ukupno</b> | <b>15</b>    | <b>74</b>              |

Prikaz metrika koda za cijelo programsko rješenje u Visual Studio okruženju:



| Hierarchy             | Maintainability Index | Cyclomatic Complexity | Depth of Inheritance | Class Coupling | Lines of Source code | Lines of Executable code |
|-----------------------|-----------------------|-----------------------|----------------------|----------------|----------------------|--------------------------|
| StudentskiDom (Debug) | 75                    | 1.400                 | 6                    | 243            | 25.804               | 6.590                    |



| Hierarchy                 | Maintainability Index | Cyclomatic Complexity | Depth of Inheritance | Class Coupling | Lines of Source code | Lines of Executable code |
|---------------------------|-----------------------|-----------------------|----------------------|----------------|----------------------|--------------------------|
| StudentskiDom (Debug)     | 75                    | 1.400                 | 6                    | 243            | 25.804               | 6.590                    |
| AppNetCore                | 79                    | 819                   | 4                    | 73             | 9.346                | 1.872                    |
| NIXConsole                | 66                    | 81                    | 4                    | 61             | 600                  | 240                      |
| StudentskiDom             | 79                    | 7                     | 1                    | 11             | 17                   | 47                       |
| StudentskiDom.Controllers | 56                    | 33                    | 3                    | 61             | 271                  | 120                      |
| StudentskiDom.Migrations  | 14                    | 44                    | 3                    | 19             | 13.006               | 4.117                    |
| StudentskiDom.Models      | 90                    | 199                   | 6                    | 72             | 1.541                | 492                      |
| Algoritam                 | 91                    | 2                     | 1                    | 2              | 41                   | 6                        |
| AzuriniMent               | 100                   | 4                     | 0                    | 2              | 7                    | 0                        |
| AzuriniStampBranica       | 100                   | 3                     | 0                    | 0              | 4                    | 0                        |
| KlasaJaga                 | 89                    | 19                    | 1                    | 12             | 12                   | 14                       |
| LisaviniMent              | 92                    | 16                    | 1                    | 11             | 70                   | 49                       |
| ErrorViewModel            | 95                    | 4                     | 1                    | 0              | 6                    | 2                        |
| GrafikaStampaPau          | 83                    | 4                     | 3                    | 11             | 24                   | 9                        |
| IKlasaJaga                | 100                   | 1                     | 0                    | 2              | 4                    | 0                        |
| Iscena                    | 100                   | 1                     | 0                    | 0              | 4                    | 0                        |
| IStudent                  | 100                   | 1                     | 0                    | 1              | 4                    | 0                        |
| IStudentskiDom            | 100                   | 3                     | 0                    | 1              | 4                    | 0                        |
| KlasaJagaMent             | 84                    | 1                     | 2                    | 11             | 29                   | 11                       |
| KlasaJagaMent             | 92                    | 14                    | 1                    | 4              | 26                   | 6                        |
| KlasaJagaMent             | 92                    | 26                    | 1                    | 6              | 37                   | 11                       |
| KlasaJagaMent             | 96                    | 9                     | 1                    | 1              | 14                   | 3                        |
| KlasaJagaMent             | 94                    | 20                    | 1                    | 7              | 40                   | 11                       |
| KlasaJagaMent             | 100                   | 1                     | 1                    | 0              | 1                    | 0                        |
| KlasaJagaMent             | 96                    | 14                    | 1                    | 2              | 22                   | 5                        |
| KlasaJagaMent             | 100                   | 3                     | 0                    | 1              | 4                    | 0                        |
| KlasaJagaMent             | 90                    | 14                    | 1                    | 4              | 46                   | 14                       |
| KlasaJagaMent             | 95                    | 0                     | 1                    | 10             | 27                   | 14                       |
| KlasaJagaMent             | 95                    | 0                     | 1                    | 16             | 36                   | 14                       |
| KlasaJagaMent             | 74                    | 4                     | 4                    | 14             | 20                   | 6                        |
| KlasaJagaMent             | 81                    | 24                    | 2                    | 16             | 102                  | 41                       |



## Verifikacija i Validacija Softvera

| Code Metrics Results        |                       |                       |                      |                |                      |                          |  |
|-----------------------------|-----------------------|-----------------------|----------------------|----------------|----------------------|--------------------------|--|
| Filter: None                |                       |                       |                      |                |                      |                          |  |
| Hierarchy                   | Maintainability Index | Cyclomatic Complexity | Depth of Inheritance | Class Coupling | Lines of Source code | Lines of Executable code |  |
| ▷ KantoniFilter             | 83                    | 5                     | 2                    | 8              | 25                   | 8                        |  |
| ▷ Korisnik                  | 93                    | 14                    | 1                    | 4              | 26                   | 6                        |  |
| ▷ LicniPodaci               | 92                    | 26                    | 1                    | 6              | 37                   | 11                       |  |
| ▷ Mjesec                    | 96                    | 9                     | 1                    | 1              | 15                   | 2                        |  |
| ▷ Paviljon                  | 94                    | 20                    | 1                    | 7              | 40                   | 8                        |  |
| ▷ Pol                       | 100                   | 1                     | 1                    | 0              | 1                    | 0                        |  |
| ▷ PrebivalisteInfo          | 96                    | 14                    | 1                    | 2              | 22                   | 3                        |  |
| ▷ PregledStanjaBonova       | 100                   | 2                     | 0                    | 1              | 5                    | 0                        |  |
| ▷ Proxy                     | 90                    | 14                    | 1                    | 4              | 36                   | 9                        |  |
| ▷ RasporedFakultet          | 55                    | 8                     | 1                    | 16             | 37                   | 14                       |  |
| ▷ RasporedKanton            | 55                    | 8                     | 1                    | 16             | 36                   | 14                       |  |
| ▷ RedovanStudent            | 74                    | 4                     | 3                    | 14             | 20                   | 6                        |  |
| ▷ Restoran                  | 81                    | 23                    | 2                    | 16             | 107                  | 31                       |  |
| ▷ Rucak                     | 96                    | 9                     | 1                    | 1              | 17                   | 2                        |  |
| ▷ SkolovanjeInfo            | 95                    | 16                    | 1                    | 2              | 24                   | 4                        |  |
| ▷ Soba                      | 92                    | 24                    | 1                    | 12             | 53                   | 13                       |  |
| ▷ StavkaNarudzbe            | 98                    | 14                    | 1                    | 1              | 24                   | 2                        |  |
| ▷ Student                   | 89                    | 35                    | 2                    | 18             | 72                   | 19                       |  |
| ▷ StudentPonovac            | 74                    | 4                     | 3                    | 14             | 20                   | 6                        |  |
| ▷ StudentskiDomContext      | 91                    | 46                    | 6                    | 29             | 58                   | 14                       |  |
| ▷ StudentskiDomSingleton    | 65                    | 42                    | 1                    | 44             | 295                  | 150                      |  |
| ▷ Uprava                    | 100                   | 5                     | 2                    | 2              | 15                   | 0                        |  |
| ▷ Vecera                    | 96                    | 9                     | 1                    | 1              | 16                   | 2                        |  |
| ▷ Zahtjev                   | 95                    | 15                    | 1                    | 6              | 28                   | 5                        |  |
| ▷ ZahtjevRestorana          | 100                   | 6                     | 2                    | 4              | 23                   | 2                        |  |
| ▷ ZahtjevStudenta           | 100                   | 6                     | 2                    | 4              | 23                   | 2                        |  |
| ▷ ZahtjevZaCimeraj          | 95                    | 20                    | 3                    | 6              | 36                   | 3                        |  |
| ▷ ZahtjevZaNabavkuNamirnica | 99                    | 4                     | 3                    | 7              | 24                   | 0                        |  |
| ▷ ZahtjevZaPremijestanje    | 99                    | 20                    | 3                    | 6              | 35                   | 1                        |  |
| ▷ ZahtjevZaUpis             | 100                   | 14                    | 2                    | 5              | 31                   | 0                        |  |
| ▷ StudentskiDom.ViewModels  | 91                    | 37                    | 1                    | 5              | 83                   | 33                       |  |

Ukupan broj pronađenih grešaka: 15

Ukupan normirani broj grešaka: 74

Broj grešaka po LOC:  $15/25804=0.0005813$

Normirani broj grešaka po LOC:  $74/25804=0.0028677$

Ukoliko je inspekcija koda trajala 5.05 sati, kolika je efikasnost otkrivanja grešaka:  
 $15/5.05=2.97$

Ukoliko je inspekcija koda trajala 2 sata, kolika je normirana efikasnost otkrivanja grešaka:  
 $15/2=7.5$

Ukoliko je normirani broj grešaka približno jednak broju grešaka bez normiranja, do kakvog se zaključka može doći?

Ukoliko je normirani broj grešaka približno jednak broju grešaka bez normiranja dalazimo do zaključka da je ozbiljnost grešaka koje su pronađene u kodu minorne.

Da li normirana efikasnost otkrivanja grešaka može biti veća od efikasnosti otkrivanja grešaka bez normiranja?

Normirana efikasnost otkrivanja grešaka može biti veća od efikasnosti otkrivanja grešaka bez normiranja

### Zadatak 5. (Analiza grešaka statističkim alatima)

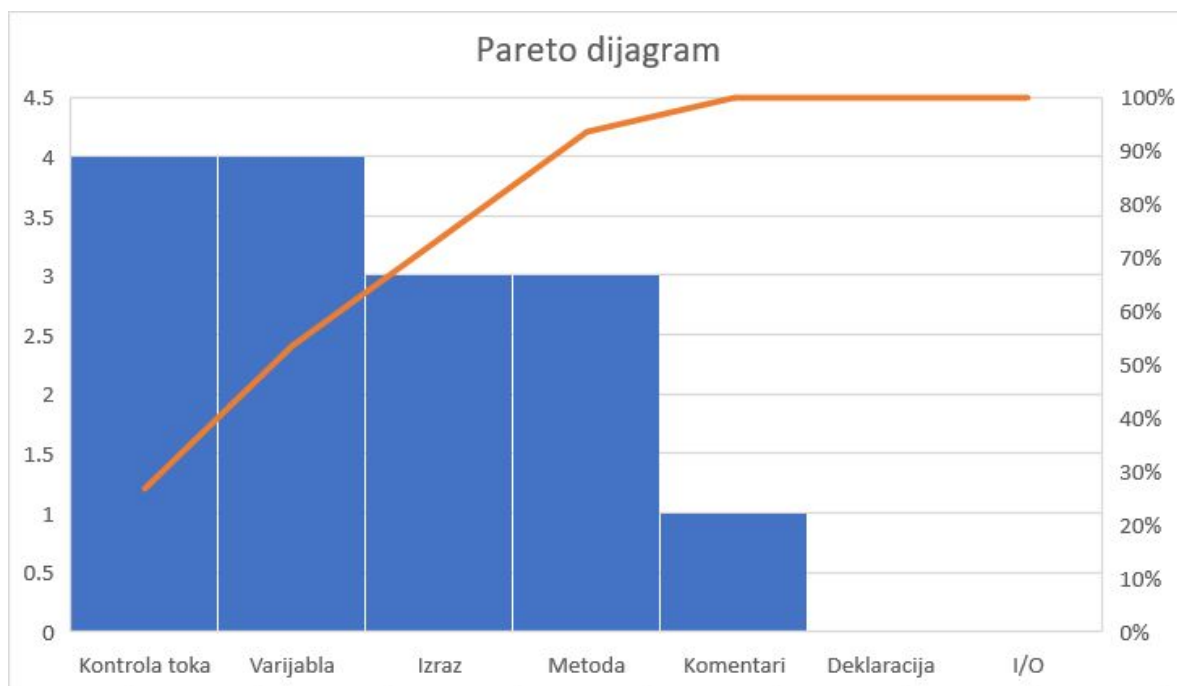
Da li se u nekom od prethodnih zadataka već koristio jedan od statističkih alata za analizu grešaka? Ukoliko je odgovor da, koji alat je u pitanju?

Da, korišten je Calculate Code Metrics (okruženje Visual Studio).

Koristeći tabelu grešaka iz zadatka 3 potrebno je napraviti Pareto dijagram grešaka. U tu svrhu prvo je potrebno napraviti sumarnu tabelu grešaka prikazanu ispod.

| Kategorija    | Broj grešaka | Kumulativni broj grešaka | Kumulativni postotak grešaka |
|---------------|--------------|--------------------------|------------------------------|
| Izraz         | 3            | 3                        | 0,2                          |
| Kontrola toka | 4            | 7                        | 0,466667                     |
| Metoda        | 3            | 10                       | 0,666667                     |
| Deklaracija   | 0            | 10                       | 0,666667                     |
| Varijabla     | 4            | 14                       | 0,933334                     |
| I/O           | 0            | 14                       | 0,933334                     |
| Komentari     | 1            | 15                       | 1                            |

Prikaz Pareto dijagrama za pronađene greške:



Šta se može zaključiti iz izgleda Pareto dijagrama? Da li je rast kumulativnog postotka linearan i šta to znači?

Iz izgleda Pareto dijagrama možemo zaključiti da najveći problem predstavljaju greške koje svrstavamo u tipove kontrola toka i varijabla. Također možemo zaključiti da je broj defekata



*Verifikacija i Validacija Softvera*

koji pripadaju tipovima kontrola toka i varijabla isti, te da je isti broj defekata koji pripadaju tipovima izraz i metoda. Rast kumulativnog postotka nije linearan, što znači da određeni tipovi grešaka imaju isti broj defekata.

*Koristeći sumarni izvještaj o greškama iz zadatka 4 potrebno je napraviti histogram ozbiljnosti grešaka.*

Prikaz histograma ozbiljnosti grešaka:



Šta se može zaključiti iz izgleda histograma? Za koju vrstu analize je pogodniji ovakav prikaz, a za koju vrstu analize je pogodnije korištenje Pareto dijagrama?

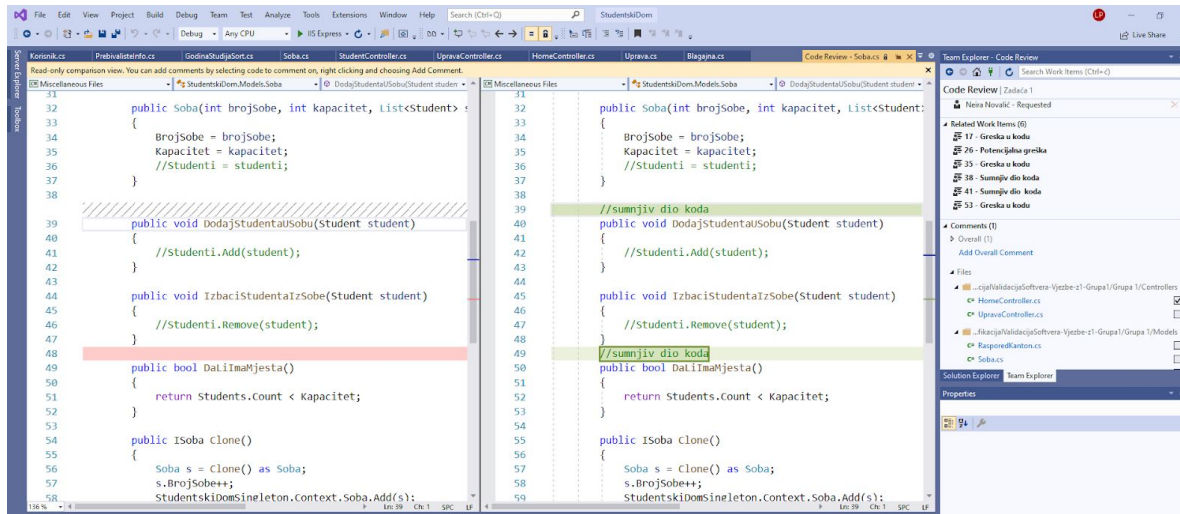
Iz izgleda histograma možemo zaključiti da se najveća frekvencija grešaka javlja kod greške tipa 3. Pareto dijagram nam daje detaljnije informacije, sortira greške po njihovoj učestalosti, te je lakše pratiti greške po prioritetu. Dok histogram daje jednostavniji prikaz, te je lakše shvatiti na koje greške treba obratiti pažnju.

*Svaki član tima treba otkloniti najmanje 3 pronađene greške u rješenju koje su im drugi članovi tima dodijelili. Nakon otklanjanja svake pojedinačne greške potrebno je zabilježiti vrijednost metrike ciklomatske kompleksnosti koja se dobiva u okviru prikaza metrika koda u Visual Studio okruženju.*

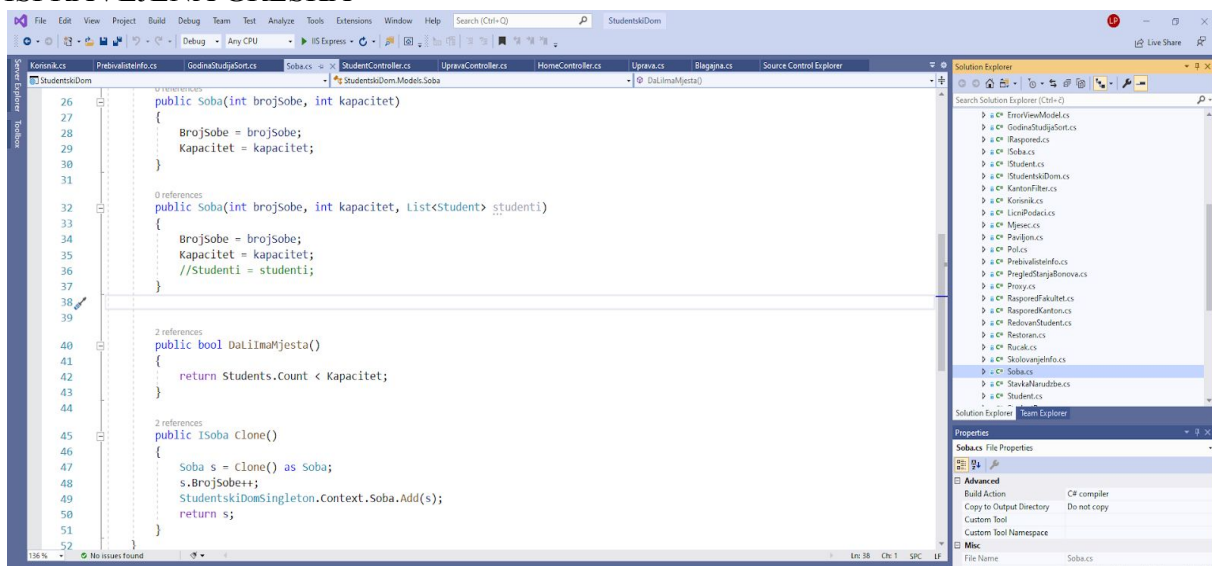
*Potrebno je prikazati izvršene promjene na način da se u okruženju prikazuje kod prije i nakon promjene. Svaki član tima u nastavku treba prikazati jedan primjer otklonjene greške.*

## Verifikacija i Validacija Softvera

### Prikaz primjera otklonjene greške (član 1): GREŠKA

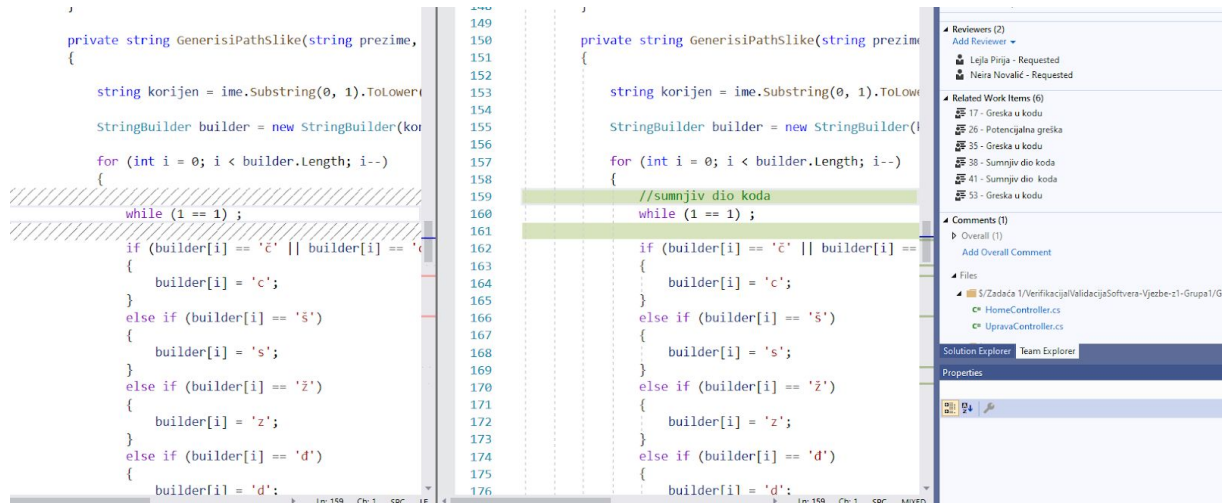


### ISPRAVLJENA GREŠKA



## Verifikacija i Validacija Softvera

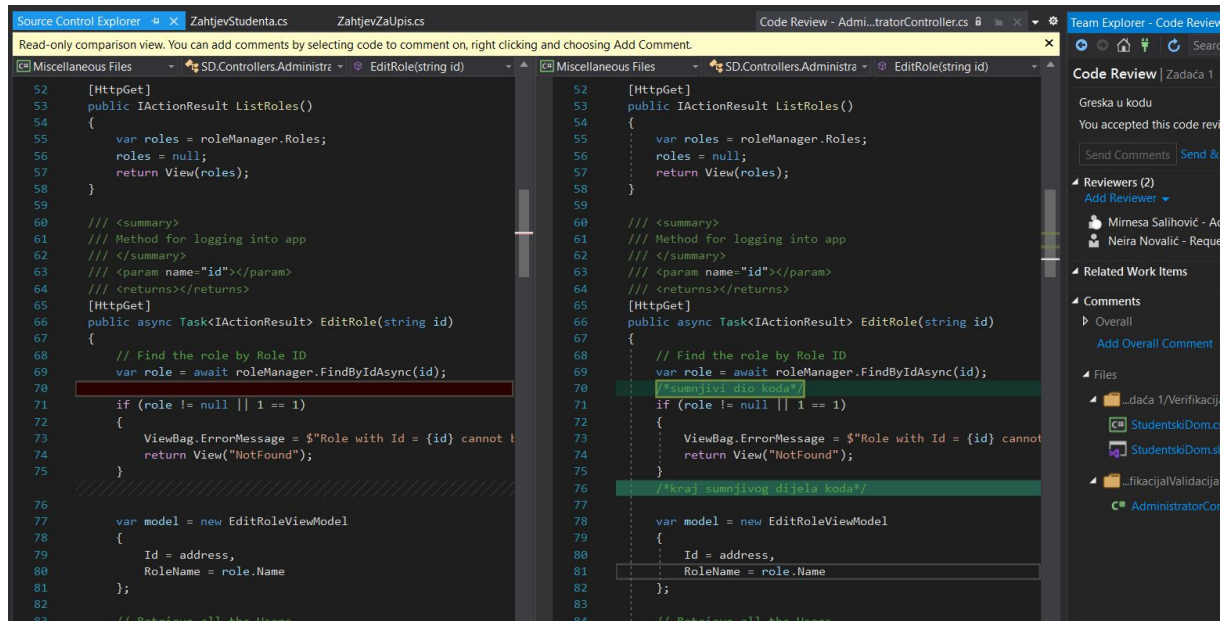
### Prikaz primjera otklonjene greške (član 2): GREŠKA



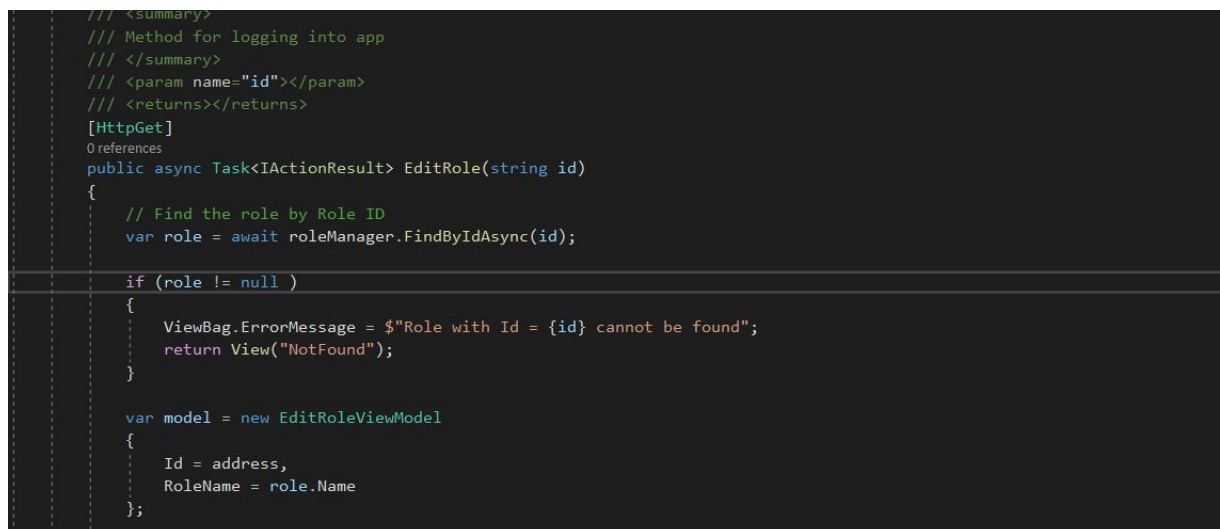
### ISPRAVLJENA GREŠKA



## Prikaz primjera otklonjene greške (član 3): GREŠKA



## ISPRAVLJENA GREŠKA



Nakon otklanjanja grešaka, potrebno je koristeći zabilježene vrijednosti metrike ciklomske kompleksnosti iz Visual Studio okruženja formirati scatter dijagram defekata.

Verifikacija i Validacija Softvera

Prikaz *scatter* dijagrama defekata:



Šta se može zaključiti iz izgleda *scatter* dijagrama? Da li se sa smanjenjem broja defekata smanjuje i kompleksnost koda? Da li to znači da je kompleksniji kod manje podložan greškama?

Scatter dijagram nam pokazuje da broj otklonjenih grešaka nije značajno uticao na sam kod jer je mali broj pravih defekata prihvaćen za uklanjanje u zadnjem dijelu zadatka. Pretežno su obrađene greške koje nemaju visok stepen ozbiljnosti. Smanjenem broja defekata se ne smanjuje kompleksnost koda i kompleksniji kod nije manje podložan greškama.