Task 2 - Inspekcija koda

Tim 2, uloge:

- Moderator i zapisničar: Selma Ljuhar
- Autor/Programer/Developer: Ajdin Šuta
- Recenzent /Ocjenjivač: svi članovi tima -> Selma Ljuhar, Ajdin Šuta, Selma Ličina, Almedina Pehlivan, Merima Durić

Contents

Tabela grešaka

Tabela 1: Klasifikacija grešaka dizajna s obzirom na ozbiljnost greške[]

Ozbiljnost greške	**Opis**
5 (kritično)	(1) Sprječava postizanje osnovnih mogućnosti. (2) Ugrožava sigurnost, zaštitu.
4	(1) Nepovoljno utiče na postizanje osnovnih funkcionalnosti, rješenje kojim se greška može izbjeći nije poznato. (2) Nepovoljno utiče na tehnički, troškovni i rasporedni rizik projekta ili sistemskog održavanja, rješenje kojim se greška može izbjeći nije poznato.
3	(1) Nepovoljno utiče na postizanje osnovnih funkcionalnosti, ali je rješenje kojim se greška može izbjeći poznato. (2) Nepovoljno utiče na tehnički, troškovni i rasporedni rizik projekta ili sistemskog održavanja, ali je rješenje kojim se greška može izbjeći poznato.
2	(1) Korisnička/operatorska neudobnost koja ne utiče na direktno na izvođenje funkcionalnosti. (2) Neugodnost za programere ili personal na održavanju, ali ne sprječava realizaciju ovih odgovornosti.
1 (minorno)	Bilo koji drugi efekat.

Opis odvijanja sesije uz naznaku alata koji se koristio

Koristili smo CodeStream. Moderator raspodijelio dijelove koda:

1. Selma Ljuhar: RequestController i StudentController
2. Almedina Pehlivan: ExamRegistrationController i HomeController
3. Ajdin Šuta: TeacherController, StudentServiceController i Models
4. Selma Ličina: CourseController i StudentCourseController
5. Merima Durić: ExamController, StudentExamController i Views

Izvještaji o greškama

- Checklista 1

Generic Checklist for Code Reviews
Structure
☑ Does the code completely and correctly implement the design?
☑ Does the code conform to any pertinent coding standards?
❌ Is the code well-structured, consistent in style, and consistently formatted?
☑ Are there any uncalled or unneeded procedures or any unreachable code?
❌ Are there any leftover stubs or test routines in the code?
☑ Can any code be replaced by calls to external reusable components or library functions?
❌ Are there any blocks of repeated code that could be condensed into a single procedure?
☑ Is storage use efficient?
❌ Are symbolics used rather than “magic number” constants or string constants?
❌ Are any modules excessively complex and should be restructured or split into multiple routines?
Documentation
☑ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
☑ Are all comments consistent with the code?
Variables
❌ Are all variables properly defined with meaningful, consistent, and clear names?
☑ Do all assigned variables have proper type consistency or casting?
☑ Are there any redundant or unused variables?
Arithmetic Operations
☑ Does the code avoid comparing floating-point numbers for equality?
☑ Does the code systematically prevent rounding errors?
☑ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
☑ Are divisors tested for zero or noise?
Loops and Branches
☑ Are all loops, branches, and logic constructs complete, correct, and properly nested?
☑ Are the most common cases tested first in IF- -ELSEIF chains?
☑ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
☑ Does every case statement have a default?
☑ Are loop termination conditions obvious and invariably achievable?
☑ Are indexes or subscripts properly initialized, just prior to the loop?
❌ Can any statements that are enclosed within loops be placed outside the loops?
☑ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?
Defensive Programming
☑ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
☑ Are imported data and input arguments tested for validity and completeness?

☑ Are all output variables assigned?
☑ Are the correct data operated on in each statement?
☑ Is every memory allocation deallocated?
❑ Are timeouts or error traps used for external device accesses?
❌ Are files checked for existence before attempting to access them?
☑ Are all files and devices are left in the correct state upon program termination?

- Izvještaj 1

Izvještaj o greškama za sesiju inspekcije

Datum(i) sesije: 11.11.2023 **Izvještaj pripremio/la:** Selma Ljuhar
Naziv projekta: Zamger
Dokument inspekcije: Pregled dizajna **Verzija:** 1
Sekcije dokumenta koje su predmet inspekcije: RequestController i StudentController
Inspekcijski tim: Selma Ličina, Ajdin Šuta, Merima Durić, Almedina Pehlivan, Selma Ljuhar

1 Lista grešaka

#	Ozbiljnost	Tip greške	Priroda greške	Opis greške	Lokacija greške
1	3	Structure	W	Pogrešan parametar u returnu	RequestController
2	1	Variables	W	CamelCase	StudentController
3	1	Variables	O	Neiskorištena varijabla	StudentController
4	1	Defensive programming	W	Moguće null reference (5)	StudentControllers
5	3	Structure	W	Loše poredan kod	RequestController
					Ukupno: 20 min

2 Aktivnosti nakon formalnog pregleda dizajna (Follow up decisions)

A	Slijede ispravke diskusiranih dijelova

3 Komentari

*W-Wrong M-Missing O-Other

- Checklista 2

Generic Checklist for Code Reviews
Structure
☑ Does the code completely and correctly implement the design?
☑ Does the code conform to any pertinent coding standards?
❌ Is the code well-structured, consistent in style, and consistently formatted?
☑ Are there any uncalled or unneeded procedures or any unreachable code?
❌ Are there any leftover stubs or test routines in the code?
❌ Can any code be replaced by calls to external reusable components or library functions?
❌ Are there any blocks of repeated code that could be condensed into a single procedure?
☑ Is storage use efficient?
❌ Are symbolics used rather than “magic number” constants or string constants?
❌ Are any modules excessively complex and should be restructured or split into multiple routines?
Documentation
☑ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
☑ Are all comments consistent with the code?
Variables
❌ Are all variables properly defined with meaningful, consistent, and clear names?
☑ Do all assigned variables have proper type consistency or casting?
☑ Are there any redundant or unused variables?
Arithmetic Operations
☑ Does the code avoid comparing floating-point numbers for equality?
☑ Does the code systematically prevent rounding errors?
☑ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
☑ Are divisors tested for zero or noise?
Loops and Branches

☑ Are all loops, branches, and logic constructs complete, correct, and properly nested?
☑ Are the most common cases tested first in IF- -ELSEIF chains?
☑ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
☑ Does every case statement have a default?
☑ Are loop termination conditions obvious and invariably achievable?
☑ Are indexes or subscripts properly initialized, just prior to the loop?
❌ Can any statements that are enclosed within loops be placed outside the loops?
☑ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?
Defensive Programming
☑ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
☑ Are imported data and input arguments tested for validity and completeness?
☑ Are all output variables assigned?
☑ Are the correct data operated on in each statement?
☑ Is every memory allocation deallocated?
❑ Are timeouts or error traps used for external device accesses?
❌ Are files checked for existence before attempting to access them?
☑ Are all files and devices are left in the correct state upon program termination?

- Izvještaj 2

Izvještaj o greškama za sesiju inspekcije

Datum(i) sesije: 11.11.2023 **Izvještaj pripremio/la:** Almedina Pehlivan
Naziv projekta: Zamger
Dokument inspekcije: Pregled dizajna **Verzija:** 1
Sekcije dokumenta koje su predmet inspekcije: ExamRegistrationController i HomeController
Inspekcijski tim: Selma Ličina, Ajdin Šuta, Merima Durić, Almedina Pehlivan, Selma Ljuhar

1 Lista grešaka

#	Ozbiljnost	Tip greške	Priroda greške	Opis greške	Lokacija greške
1	1	Structure	O	Nepotreban kod	ExamRegistrationController
2	1	Defensive programming	W	Moguća null referenca	HomeController
3	2	Structure	O	Dead code	HomeController
4	2	Structure	O	Neiskorištena funkcija	ExamRegistrationController

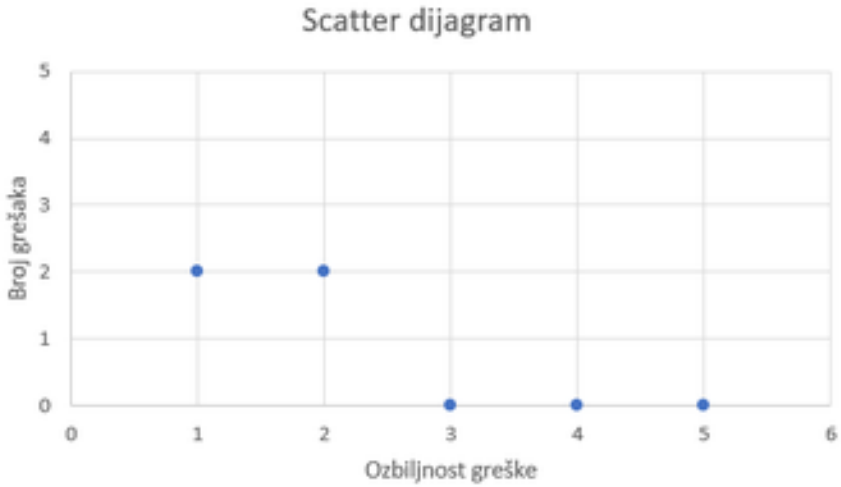
	Ukupno: 15 min

2 Aktivnosti nakon formalnog pregleda dizajna (Follow up decisions)

A	Slijede ispravke diskusiranih dijelova

3 Komentari

*W-Wrong M-Missing O-Other



- Checklista 3

Generic Checklist for Code Reviews
Structure
☑ Does the code completely and correctly implement the design?
☑ Does the code conform to any pertinent coding standards?
☑ Is the code well-structured, consistent in style, and consistently formatted?
☑ Are there any uncalled or unneeded procedures or any unreachable code?
❌ Are there any leftover stubs or test routines in the code?
❌ Can any code be replaced by calls to external reusable components or library functions?
❌ Are there any blocks of repeated code that could be condensed into a single procedure?
☑ Is storage use efficient?
❌ Are symbolics used rather than “magic number” constants or string constants?
❌ Are any modules excessively complex and should be restructured or split into multiple routines?
Documentation
☑ Is the code clearly and adequately documented with an easy-to-maintain commenting style?

☑ Are all comments consistent with the code?
Variables
☑ Are all variables properly defined with meaningful, consistent, and clear names?
☑ Do all assigned variables have proper type consistency or casting?
☑ Are there any redundant or unused variables?
Arithmetic Operations
☑ Does the code avoid comparing floating-point numbers for equality?
☑ Does the code systematically prevent rounding errors?
☑ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
☑ Are divisors tested for zero or noise?
Loops and Branches
☑ Are all loops, branches, and logic constructs complete, correct, and properly nested?
☑ Are the most common cases tested first in IF- -ELSEIF chains?
☑ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
☑ Does every case statement have a default?
☑ Are loop termination conditions obvious and invariably achievable?
☑ Are indexes or subscripts properly initialized, just prior to the loop?
❌ Can any statements that are enclosed within loops be placed outside the loops?
☑ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?
Defensive Programming
☑ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
☑ Are imported data and input arguments tested for validity and completeness?
☑ Are all output variables assigned?
☑ Are the correct data operated on in each statement?
☑ Is every memory allocation deallocated?
❑ Are timeouts or error traps used for external device accesses?
☑ Are files checked for existence before attempting to access them?
☑ Are all files and devices are left in the correct state upon program termination?

- Izvještaj 3

Izvještaj o greškama za sesiju inspekcije

Datum(i) sesije: 11.11.2023 **Izvještaj pripremio/la:** Ajdin Šuta
Naziv projekta: Zamger
Dokument inspekcije: Pregled dizajna **Verzija:** 1
Sekcije dokumenta koje su predmet inspekcije: TeacherController, StudentServiceController i Models
Inspekcijski tim: Selma Ličina, Ajdin Šuta, Merima Durić, Almedina Pehlivan, Selma Ljuhar

1 Lista grešaka

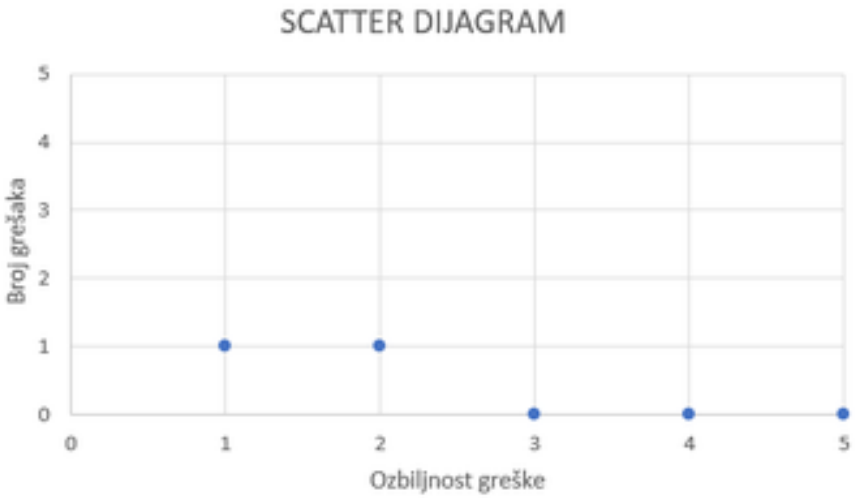
#	Ozbiljnost	Tip greške	Priroda greške	Opis greške	Lokacija greške
1	1	Structure	O	Nepotreban kod	StudentServiceController
2	2	Structure	O	Kod bez funkcionalnosti	StudentCourseManager
					Ukupno: 20 min

2 Aktivnosti nakon formalnog pregleda dizajna (Follow up decisions)

A	Slijede ispravke diskusiranih dijelova

3 Komentari

*W-Wrong M-Missing O-Other



- Checklista 4

Generic Checklist for Code Reviews
Structure
☑ Does the code completely and correctly implement the design?
☑ Does the code conform to any pertinent coding standards?
☑ Is the code well-structured, consistent in style, and consistently formatted?
☑ Are there any uncalled or unneeded procedures or any unreachable code?
❌ Are there any leftover stubs or test routines in the code?
☑ Can any code be replaced by calls to external reusable components or library functions?

❌ Are there any blocks of repeated code that could be condensed into a single procedure?
☑ Is storage use efficient?
❌ Are symbolics used rather than “magic number” constants or string constants?
❌ Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

☑ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
☑ Are all comments consistent with the code?

Variables

❌ Are all variables properly defined with meaningful, consistent, and clear names?
☑ Do all assigned variables have proper type consistency or casting?
☑ Are there any redundant or unused variables?

Arithmetic Operations

☑ Does the code avoid comparing floating-point numbers for equality?
☑ Does the code systematically prevent rounding errors?
☑ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
☑ Are divisors tested for zero or noise?

Loops and Branches

☑ Are all loops, branches, and logic constructs complete, correct, and properly nested?
☑ Are the most common cases tested first in IF- -ELSEIF chains?
☑ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
☑ Does every case statement have a default?
☑ Are loop termination conditions obvious and invariably achievable?
☑ Are indexes or subscripts properly initialized, just prior to the loop?
❌ Can any statements that are enclosed within loops be placed outside the loops?
☑ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Defensive Programming

☑ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
☑ Are imported data and input arguments tested for validity and completeness?
☑ Are all output variables assigned?
☑ Are the correct data operated on in each statement?
☑ Is every memory allocation deallocated?
❑ Are timeouts or error traps used for external device accesses?
❌ Are files checked for existence before attempting to access them?
☑ Are all files and devices are left in the correct state upon program termination?

- Izvještaj 4

Izvještaj o greškama za sesiju inspekcije

Datum(i) sesije: 11.11.2023 **Izvještaj pripremio/la:** Selma Ličina
Naziv projekta: Zamger
Dokument inspekcije: Pregled dizajna **Verzija:** 1
Sekcije dokumenta koje su predmet inspekcije: CourseController i StudentCourseController
Inspekcijski tim: Selma Ličina, Ajdin Šuta, Merima Durić, Almedina Pehlivan, Selma Ljuhar

1 Lista grešaka

#	Ozbiljnost	Tip greške	Priroda greške	Opis greške	Lokacija greške
1	1	Variables	W	CamelCase	CourseController
2	1	Structure	O	Potrebna optimizacija	CourseController
3	2	Structure	W	Netačan i nepotreban kod	CourseController
4	1	Defensive programming	W	Moguće null reference (2)	StudentCourseController
					Ukupno: 15 min

2 Aktivnosti nakon formalnog pregleda dizajna (Follow up decisions)

A	Slijede ispravke diskusiranih dijelova

3 Komentari

*W-Wrong M-Missing O-Other

- Checklista 5

Generic Checklist for Code Reviews
Structure
☑ Does the code completely and correctly implement the design?
☑ Does the code conform to any pertinent coding standards?
☑ Is the code well-structured, consistent in style, and consistently formatted?
❌ Are there any uncalled or unneeded procedures or any unreachable code?
❌ Are there any leftover stubs or test routines in the code?
❌ Can any code be replaced by calls to external reusable components or library functions?
❌ Are there any blocks of repeated code that could be condensed into a single procedure?
☑ Is storage use efficient?
❌ Are symbolics used rather than “magic number” constants or string constants?
❌ Are any modules excessively complex and should be restructured or split into multiple routines?
Documentation
☑ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
☑ Are all comments consistent with the code?
Variables
☑ Are all variables properly defined with meaningful, consistent, and clear names?
☑ Do all assigned variables have proper type consistency or casting?
☑ Are there any redundant or unused variables?
Arithmetic Operations
☑ Does the code avoid comparing floating-point numbers for equality?
☑ Does the code systematically prevent rounding errors?
☑ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
☑ Are divisors tested for zero or noise?
Loops and Branches
☑ Are all loops, branches, and logic constructs complete, correct, and properly nested?
☑ Are the most common cases tested first in IF- -ELSEIF chains?
☑ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
☑ Does every case statement have a default?
☑ Are loop termination conditions obvious and invariably achievable?
☑ Are indexes or subscripts properly initialized, just prior to the loop?
❌ Can any statements that are enclosed within loops be placed outside the loops?
☑ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?
Defensive Programming
☑ Are indexes, pointers, and subscripts tested against array, record, or file bounds?

☑ Are imported data and input arguments tested for validity and completeness?
☑ Are all output variables assigned?
☑ Are the correct data operated on in each statement?
☑ Is every memory allocation deallocated?
❑ Are timeouts or error traps used for external device accesses?
☑ Are files checked for existence before attempting to access them?
☑ Are all files and devices are left in the correct state upon program termination?

- Izvještaj 5

Izvještaj o greškama za sesiju inspekcije

Datum(i) sesije: 11.11.2023 **Izvještaj pripremio/la:** Merima Durić
Naziv projekta: Zamger
Dokument inspekcije: Pregled dizajna **Verzija:** 1
Sekcije dokumenta koje su predmet inspekcije: ExamController, StudentExamController i Views
Inspekcijski tim: Selma Ličina, Ajdin Šuta, Merima Durić, Almedina Pehlivan, Selma Ljuhar

1 Lista grešaka

#	Ozbiljnost	Tip greške	Priroda greške	Opis greške	Lokacija greške
1	4	Structure	W	Nefunkcionalan kod	Views/Home
					Ukupno: 20 min

2 Aktivnosti nakon formalnog pregleda dizajna (Follow up decisions)

A	Slijede ispravke diskusiranih dijelova

3 Komentari

*W-Wrong M-Missing O-Other

Sumarni izvještaj

Sumarni izvještaj za sesiju inspekcije

Datum sesije: 11.11.2023
Naziv projekta: Zamger
Dokument inspekcije: Detaljni dizajn **Verzija:** 2
Ukupno: (A) 22 stranice/linije tekst
Inspekcijski tim: Selma Ličina, Ajdin Šuta, Merima Durić, Almedina Pehlivan,

Sumarno o greškama

Ozbiljnost greške	Prirode greške W	M	O	Ukupno grešaka	Faktor ozbiljnosti	Ukupno grešaka (standarizovano)	Komentar
5 - kritična				0	16	0	/
4	1			1	8	8	/
3	2			2	4	8	/
2	1		3	4	2	8	/
1 - minorna	5		4	9	1	9	/
Ukupno	9	0	7	(C)16		(D)33	

Metrike

(1) Prosječno grešaka po stranici = C/A = 16/22 = 0.7272
(2) Prosječno grešaka po stranici (standardizovano) = D/A = 33/22 = 1.5
(3) Efikasnost detekcije grešaka (sati po greškama) = B/C = 1.5/16 = 0.09375
(4) Efikasnost detekcije grešaka (standardizovano) = B/D = 1.5/33 = 0.045

Spisak urađenih korektivnih akcija

1. Prolazak redom kroz kod i resolvanje prvo lakših grešaka i komentara.
2. Resolvanje issues oko null referenci i grešaka veće ozbiljnosti.
3. Provjera da li program radi kako treba.
4. Moderator nakon provjere merga promjene u main.

- Prikaz korektivnih akcija:

Promjene koda u CourseController(crvene linije obrisano, zelene kod poslije promjene):

```
24  24
25  25          public List<SelectListItem>  GetTeacherNamesList()
26  26          {
27      -           List<SelectListItem> Teachers = new List<SelectListItem>();
    27  +           List<SelectListItem> Teachers = new();
28  28
29  29              foreach (var item in _context.Teacher)
30  30              {
        @@ -34,7 +34,7 @@ public List<SelectListItem>  GetTeacherNamesList()
34  34              return Teachers;
35  35
36  36          }
37      -       public List<Course> selectionSort(List<Course> Index)
    37  +       public List<Course> SelectionSort(List<Course> Index)
38  38          {
39  39              for (int i = 0; i < Index.Count - 1; i++)
40  40              {
        @@ -48,9 +48,7 @@ public List<Course> selectionSort(List<Course> Index)
48  48                  }
49  49                  if (minIndex != i)
50  50                  {
51      -                   Course temp = Index[i];
52      -                   Index[i] = Index[minIndex];
53      -                   Index[minIndex] = temp;
    51  +                   (Index[minIndex], Index[i]) = (Index[i], Index[minIndex]);
54  52                  }
55  53              }
56  54              return Index;
        @@ -59,7 +57,7 @@ public List<Course> selectionSort(List<Course> Index)
59  57          public async Task<IActionResult> Index()
60  58          {
61  59              var applicationDbContext = await _context.Course.Include(c => c.Teacher).ToListAsync();
62      -           applicationDbContext = selectionSort(applicationDbContext);
    60  +           applicationDbContext = SelectionSort(applicationDbContext);
```

```
62        -             applicationDbContext = selectionSort(applicationDbContext);
      60  +             applicationDbContext = SelectionSort(applicationDbContext);
63    61
64    62
65    63              return View(applicationDbContext);
          @@ -208,28 +206,6 @@ public async Task<IActionResult> CourseStatus(int? id)
208   206             var course = await _context.Course.FindAsync(id);
209   207             ViewData["course"] = course;
210   208             ViewData["Courses"] = courses;
211       -             /*
212       -
213       -             var StudentCourses = await _context.StudentCourse.Where(sc => sc.Course == course).ToListAsync();
214       -             var Students = new List<Student>();
215       -             var user = await _userManager.GetUserAsync(User);
216       -             var Courses = await _context.Course.Where(c => c.Teacher == user).ToListAsync();
217       -             foreach (var item in StudentCourses)
218       -             {
219       -                 Students.Add(item.Student);
220       -             }
221       -
222       -             var Exams = await _context.Exam.Where(e => e.Course == course).ToListAsync();
223       -
224       -             var Homeworks = await _context.Homework.Where(h => h.Course == course).ToListAsync();
225       -
226       -
227       -             ViewData["Courses"] = Courses;
228       -             ViewData["StudentCourses"] = StudentCourses;
229       -             ViewData["Students"] = Students;
230       -             ViewData["Exams"] = Exams;
231       -             ViewData["Homeworks"] = Homeworks;
232       -             */
233   209             var students = await _context.StudentCourse.Where(sc => sc.Course == course).ToListAsync();
234   210             List<StudentCourseInfo> list = await _courseManager.RetrieveStudentCourseInfo(id);
235   211             ViewData["Info"] = list;
```

Nije bila ispoštovana konvencija o imenovanju varijabli i funkcija, obrisani su bespotrebni komentari, uvodi se korištenje kraće sintakse za operator “new” koja je uvedena u novijim verzijama C# i izostavlja suvišne deklaracije tipa kada se tip može zaključiti iz konteksta, uvodi direktna zamjena vrijednosti na pozicijama i i minIndex, bez potrebe za privremenom promjenljivom temp.

Promjene koda u ExamRegistrationController:

5 Implementacija/Controllers/ExamRegistrationController.cs

```
@@ -1,7 +1,6 @@
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
- using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using ooadproject.Data;
using ooadproject.Models;
@@ -61,10 +60,6 @@ public async Task<IActionResult> Create(int examID, [Bind("StudentID,ExamID,Regi
                    return RedirectToAction(nameof(Index));
                }
-
-            // ViewData["ExamID"] = new SelectList(_context.Exam, "ID", "ID", examRegistration.ExamID);
-             //ViewData["StudentID"] = new SelectList(_context.Student, "Id", "Id", examRegistration.StudentID);
-
                return View(examRegistration);
            }
```

Obrisani bespotrebni komentari i usings.

Promjene koda u HomeController:

26 Implementacija/Controllers/HomeController.cs

```
@@ -69,7 +69,17 @@ public async Task<IActionResult> TeacherHome()
            var teacher = await _userManager.GetUserAsync(User);

            var Courses = await _context.Course.Where(c => c.TeacherID == teacher.Id).ToListAsync();
-           var Exams = await _context.Exam.Include(e => e.Course).Where(e => e.Course.TeacherID == teacher.Id && e.Time > DateTime.Now).ToListAsync();
+           var Exams = new List<Exam>();
+           var exams = await _context.Exam.Include(e => e.Course).ToListAsync();
+           foreach (var exam in exams)
+           {
+               if (exam.Course == null)
+                   continue;
+               if (exam.Course.TeacherID == teacher.Id && exam.Time > DateTime.Now)
+               {
+                   Exams.Add(exam);
+               }
+           }
            var registered = new Dictionary<int, int>();
            foreach (var item in Exams)
            {
@@ -84,14 +94,22 @@ public async Task<IActionResult> TeacherHome()
        }

-       // nije implement, kopirano TeacherHome
        public async Task<IActionResult> StudentServiceHome()
        {
            var teacher = await _userManager.GetUserAsync(User);

            var Courses = await _context.Course.Where(c => c.TeacherID == teacher.Id).ToListAsync();
-           var Exams = await _context.Exam.Include(e => e.Course).Where(e => e.Course.TeacherID == teacher.Id && e.Time > DateTime.Now).ToListAsync();
-
+           var Exams = new List<Exam>();
+           var exams = await _context.Exam.Include(e => e.Course).ToListAsync();
+           foreach (var exam in exams)
+           {
+               if (exam.Course == null)
+                   continue;
+               if (exam.Course.TeacherID == teacher.Id && exam.Time > DateTime.Now)
+               {
+                   Exams.Add(exam);
+               }
+           }
            ViewData["Courses"] = Courses;
            ViewData["Exams"] = Exams;
```

Ispravljena implementacija StudentServiceHome i TeacherHome, obrisan komentar

Promjene koda u RequestController:

13 Implementacija/Controllers/RequestController.cs

```
@@ -1,9 +1,4 @@
- using System;
- using System.Collections.Generic;
- using System.Linq;
- using System.Runtime.CompilerServices;
- using System.Threading.Tasks;
- using Microsoft.AspNetCore.Authorization;
+ using Microsoft.AspNetCore.Authorization;
  using Microsoft.AspNetCore.Identity;
  using Microsoft.AspNetCore.Mvc;
  using Microsoft.AspNetCore.Mvc.Rendering;
@@ -26,7 +21,7 @@ public RequestController(ApplicationDbContext context, UserManager<Person> userM
          public List<SelectListItem> GetRequestTypesList()
          {
-             List<SelectListItem> Types = new List<SelectListItem>();
+             List<SelectListItem> Types = new();
              var EnumValues = Enum.GetValues(typeof(RequestType));

              foreach (var value in EnumValues)
@@ -44,7 +39,7 @@ public List<SelectListItem> GetRequestTypesList()
          public List<SelectListItem> GetRequestStatusList()
          {
              //Get all request types from enum of model RequestStatus
-             List<SelectListItem> Types = new List<SelectListItem>();
+             List<SelectListItem> Types = new();
              var EnumValues = Enum.GetValues(typeof(RequestStatus));
              foreach (var value in EnumValues)
              {
@@ -191,7 +186,7 @@ public async Task<IActionResult> Edit(int id, [Bind("ID,RequesterID,RequestTime,
                  return RedirectToAction(nameof(Index));
              }

-             return View(Index);
+             return View(request);
          }
```

Uvodi se korištenje kraće sintakse za operator “new” koja je uvedena u novijim verzijama C# i izostavlja suvišne deklaracije tipa kada se tip može zaključiti iz konteksta, sada se request prosleđuje View-u kako bi se prikazali podaci koji su izmijenjeni ili ažurirani.

Promjene koda u StudentController:

19 Implementacija/Controllers/StudentController.cs

```
@@ -1,13 +1,6 @@
- using System;
- using System.Collections.Generic;
- using System.Linq;
- using System.Threading.Tasks;
-
- using Microsoft.AspNetCore.Authorization;
+ using Microsoft.AspNetCore.Authorization;
  using Microsoft.AspNetCore.Identity;
-
  using Microsoft.AspNetCore.Mvc;
- using Microsoft.AspNetCore.Mvc.Rendering;
  using Microsoft.EntityFrameworkCore;
  using ooadproject.Data;
  using ooadproject.Models;
@@ -29,7 +22,7 @@ public StudentController(ApplicationDbContext context, UserManager<Person> userM
          }


-         public List<Student> bubbleSort(List<Student> Index)
+         public List<Student> BubbleSort(List<Student> Index)
          {
              //Bubble sort of index by name
              for (int i = 0; i < Index.Count - 1; i++)
@@ -38,9 +31,7 @@ public List<Student> bubbleSort(List<Student> Index)
                  {
                      if (Index[j].FirstName.CompareTo(Index[j + 1].FirstName) > 0)
                      {
-                         var temp = Index[j];
-                         Index[j] = Index[j + 1];
-                         Index[j + 1] = temp;
+                         (Index[j + 1], Index[j]) = (Index[j], Index[j + 1]);
                      }
                  }
              }
```

Promjena imenovanja funkcija da bi se ispoštovala konvencija, obrisani su bespotrebni usings I dodani oni koji nedostaju, optimizirana zamjena varijabli (bez uvođenj temp varijabli)

StudentController

```
@@ -50,7 +41,7 @@ public async Task<IActionResult> Index()
50  41          {
51  42              var Index = await _context.Student.ToListAsync();
52  43
53      -           Index = bubbleSort(Index);
    44  +           Index = BubbleSort(Index);
54  45
55  46              return _context.Student != null ?
56  47                          View(Index) :
@@ -133,7 +124,7 @@ public async Task<IActionResult> Edit(int id, [Bind("Index,Department,Year,Id,Fi
133 124                     _context.Update(student);
134 125                     await _context.SaveChangesAsync();
135 126                 }
136     -               catch (DbUpdateConcurrencyException ex)
    127 +               catch (DbUpdateConcurrencyException)
137 128                 {
138 129                     if (!StudentExists(student.Id))
139 130                     {
```

Promjena imenovanja funkcije da se ispoštuje konvencija i optimizacija try-catch bloka

Promjena koda u StudentCourseController:

32 Implementacija/Controllers/StudentCourseController.cs

```
... ...  @@ -1,8 +1,4 @@
1       - using System;
2       - using System.Collections.Generic;
3       - using System.Linq;
4       - using System.Threading.Tasks;
5       - using Microsoft.AspNetCore.Mvc;
    1   + using Microsoft.AspNetCore.Mvc;
6   2     using Microsoft.AspNetCore.Mvc.Rendering;
7   3     using Microsoft.EntityFrameworkCore;
8   4     using ooadproject.Data;
@@ -118,13 +114,18 @@ public async Task<IActionResult> StudentCourseStatus(int? id)
118 114             var user = await _userManager.GetUserAsync(User);
119 115             var courses = await _context.StudentCourse.Include(sc => sc.Course).Where(sc => sc.StudentID == user.Id).ToListAsync();
120 116             var StudentCourse = await _context.StudentCourse.FindAsync(id);
121     -           StudentCourse.Course.Teacher = await _context.Teacher.FindAsync(StudentCourse.Course.TeacherID);
    117 +           if (StudentCourse != null && StudentCourse.Course != null) {
    118 +               StudentCourse.Course.Teacher = await _context.Teacher.FindAsync(StudentCourse.Course.TeacherID);
    119 +           }
122 120             //Set Teacher for every StudentCourse.Course
123 121             foreach (var item in courses)
124 122             {
125     -               item.Course.Teacher = await _context.Teacher.FindAsync(item.Course.TeacherID);
    123 +               if (item.Course != null)
    124 +               {
    125 +                   item.Course.Teacher = await _context.Teacher.FindAsync(item.Course.TeacherID);
    126 +               }
126 127             }
127     -
    128 +
128 129             var StudentExams = await _context.StudentExam.Where(se => se.CourseID == id).ToListAsync();
129 130             var StudentHomeworks = await _context.StudentHomework.Where(sh => sh.CourseID == id).ToListAsync();
130 131
```

Obrisani bespotrebni usings, dodana provjera da li su objekti null da bi se spriječilo pristupanje neincijaliziranim objektima.

```
@@ -133,13 +134,13 @@ public async Task<IActionResult> StudentCourseStatus(int? id)
133 134             var studentExams = await _context.StudentExam
134 135             .Include(se => se.Exam)
135 136             .Where(se => se.CourseID == id)
136     -           .Select(se => new { se.PointsScored, se.Exam.TotalPoints })
    137 +           .Select(se => new { se.PointsScored, se.Exam!.TotalPoints })
137 138             .ToListAsync();
138 139
139 140             var studentHomeworks = await _context.StudentHomework
140 141                 .Include(sh => sh.Homework)
141 142                 .Where(sh => sh.CourseID == id)
142     -           .Select(sh => new { sh.PointsScored, sh.Homework.TotalPoints })
    143 +           .Select(sh => new { sh.PointsScored, sh.Homework!.TotalPoints })
143 144             .ToListAsync();
144 145
145 146             double scored = studentExams.Sum(se => se.PointsScored) + studentHomeworks.Sum(sh => sh.PointsScored);

@@ -185,7 +186,16 @@ public async Task<IActionResult> StudentOverallStatus(int? id)
185 186                     CoursesWithGrade.Add(item);
186 187                 }
187 188             }
188     -           ViewData["GradedCourses"] = CoursesWithGrade.OrderByDescending(c => c.Course.Semester).ThenBy(c => c.Course.Name).ToList();
    189 +
    190 +           ViewData["GradedCourses"] = CoursesWithGrade.OrderByDescending(c =>
    191 +           {
    192 +               return c.Course!.Semester;
    193 +           }).ThenBy(c =>
    194 +           {
    195 +               return c.Course?.Name;
    196 +           })
    197 +           .ToList();
    198 +
189 199             ViewData["Courses"] = courses;
190 200             //Calculate the average grade for all courses
191 201             double AverageGrade = 0;
```

Dodan operator 'null propogation' (!), koji se koristi da bi se naglasilo da se zna da je Exam(Homework) referenca sigurno nema null vrijednost, uvodi se operator ? da se sugeriše da to svojstvo može biti null

Promjene koda u StudentExamController:

13 Implementacija/Controllers/StudentExamController.cs

```
@@ -1,8 +1,4 @@
- using System;
- using System.Collections.Generic;
- using System.Linq;
- using System.Threading.Tasks;
- using Microsoft.AspNetCore.Mvc;
+ using Microsoft.AspNetCore.Mvc;
  using Microsoft.AspNetCore.Mvc.Rendering;
  using Microsoft.EntityFrameworkCore;
  using ooadproject.Data;

@@ -40,7 +36,7 @@ public async Task<IActionResult> TeacherInput(int id)
              var user = await _userManager.GetUserAsync(User);
              var courses = await _context.Course.Where(c => c.TeacherID == user.Id).ToListAsync();
              var currentCourse = await _context.Course.FindAsync(id);
-             var exams = await _context.Exam.Include(e => e.Course).Where(e => courses.Contains(e.Course)).ToListAsync();
+             var exams = await _context.Exam.Include(e => e.Course).Where(e => courses.Contains(e.Course!)).ToListAsync();
              ViewData["Exams"] = new SelectList(exams,"ID","Type");
              ViewData["CurrentCourse"] = currentCourse;
              ViewData["Courses"] = courses;

@@ -85,7 +81,10 @@ public List<SelectListItem> GetFullNames(List<StudentCourse> owo)

              foreach (StudentCourse item in owo)
              {
-                 Students.Add(new SelectListItem() { Text = $"{item.Student.FirstName} {item.Student.LastName}", Value = item.ID.ToString() });
+                 if (item.Student != null)
+                 {
+                     Students.Add(new SelectListItem() { Text = $"{item.Student.FirstName} {item.Student.LastName}", Value = item.ID.ToString() });
+                 }
              }

              return Students;
```

Obrisani su bespotrebni usings, dodan operator ! (objasnjeno u prethodnom primjeru), dodana provjera za null vrijednosti

Promjene koda u StudentServiceController:

4 Implementacija/Controllers/StudentServiceController.cs

```
@@ -11,10 +11,10 @@ namespace ooadproject.Controllers
      public class StudentServiceController : Controller
      {
          private readonly ApplicationDbContext _context;
-         private readonly Microsoft.AspNetCore.Identity.UserManager<Person> _userManager;
+         private readonly UserManager<Person> _userManager;
          private readonly IPasswordHasher<Person> _passwordHasher;

-         public StudentServiceController(ApplicationDbContext context, Microsoft.AspNetCore.Identity.UserManager<Person> userManager, IPasswordHasher<Person> passwordHasher)
+         public StudentServiceController(ApplicationDbContext context, UserManager<Person> userManager, IPasswordHasher<Person> passwordHasher)
          {
              _context = context;
              _userManager = userManager;
```

Uvodi se korištenje UserManager<Person> jer je lakše za održavanje i poboljšava čitljivost koda u poređenju sa Microsoft.AspNetCore.Identity.UserManager<Person>

Nakon provjere ispravnosti koda, urađen je merge na main branch.