Selma Ljuhar 19177

White box testiranje – izvještaj

Metoda StudentServiceHome() u HomeController-u:

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

    return View();
}
```

1. Line coverage

```
[TestMethod]
public void StudentServiceHome_LineCoverageTest()
{

   _userManagerMock.Setup(m => m.GetUserAsync(It.IsAny<ClaimsPrincipal>())).ReturnsAsync(teacher);
   var result = await controller.StudentServiceHome();
   Assert.IsNotNull(result);
   Assert.IsInstanceOfType(result, typeof(ViewResult));
   Assert.IsTrue(viewResult.ViewData.ContainsKey("Courses"));
   Assert.IsTrue(viewResult.ViewData.ContainsKey("Exams"));
   Assert.AreEqual(courses.Count, ((List<Course>)viewResult.ViewData["Courses"]).Count);
   Assert.AreEqual(exams.Count, ((List<Exam>)viewResult.ViewData["Exams"]).Count);
   Assert.IsInstanceOfType(viewResult.ViewData["Courses"], typeof(List<Course>));
   Assert.IsInstanceOfType(viewResult.ViewData["Exams"], typeof(List<Exam>));
}
```

2. Branch coverage

   Dovoljno je postaviti dva različita datuma, jedan koji ispunjava i jedan koji ne ispunjava uslov.

```
[TestMethod]
public void StudentServiceHome_BranchCoverageTest()
{
   _userManagerMock.Setup(m => m.GetUserAsync(It.IsAny<ClaimsPrincipal>())).ReturnsAsync(teacher);
   var exams = new List<Exam>
   {
      new Exam { Course = new Course { TeacherID = 1 }, Time = DateTime.Now.AddDays(1) },
      new Exam { Course = new Course { TeacherID = 2 }, Time = DateTime.Now.AddDays(-1) }
   };
   _contextMock.Setup(c => c.Exam.Include(It.IsAny<Expression<Func<Exam, object>>>()))
      .Returns(exams.AsQueryable());
   var result = await controller.StudentServiceHome() as ViewResult;
   Assert.IsNotNull(result);
   Assert.IsTrue(result.ViewData.ContainsKey("Exams"));
   var examsInViewData = result.ViewData["Exams"] as List<Exam>;
   Assert.IsNotNull(examsInViewData);
   Assert.AreEqual(1, examsInViewData.Count);
   Assert.AreEqual(exams[0], examsInViewData[0]);
}
```

3. Condition coverage

   Potrebno ispitati sve kombinacije datuma i id-ova Teachera, kad oba zadovoljavaju, samo po jedan od ta dva zadovoljava i kad nijedan ne zadovoljava uslov.

```
[TestMethod]
public void StudentServiceHome_ConditionCoverageTest()
{
   _userManagerMock.Setup(m => m.GetUserAsync(It.IsAny<ClaimsPrincipal>())).ReturnsAsync(teacher);
   var exams = new List<Exam>
   {
      new Exam { Course = new Course { TeacherID = teacher.Id }, Time = DateTime.Now.AddDays(1) },
      new Exam { Course = new Course { TeacherID = 9999 }, Time = DateTime.Now.AddDays(1) },
      new Exam { Course = new Course { TeacherID = teacher.Id }, Time = DateTime.Now.AddDays(-1) },
```

```
        new Exam { Course = new Course { TeacherID = 9999}, Time = DateTime.Now.AddDays(-1) }
    };
    _contextMock.Setup(c => c.Exam.Include(It.IsAny<Expression<Func<Exam, object>>>()))
        .Returns(exams.AsQueryable());
    var result = await controller.StudentServiceHome() as ViewResult;
    Assert.IsNotNull(result);
    Assert.IsTrue(result.ViewData.ContainsKey("Exams"));
    var examsInViewData = result.ViewData["Exams"] as List<Exam>;
    Assert.IsNotNull(examsInViewData);
    Assert.AreEqual(1, examsInViewData.Count);
    Assert.AreEqual(exams[0], examsInViewData[0]);
}
```

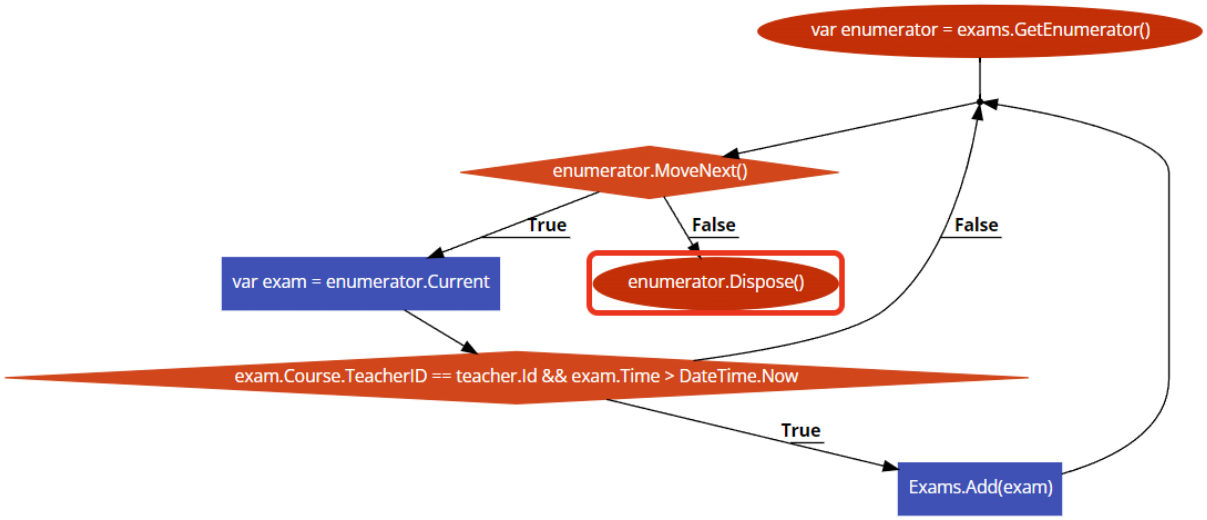
4. Loop coverage

    Provjera iteracija na osnovu dodavanja broja dodanih Exam-a.

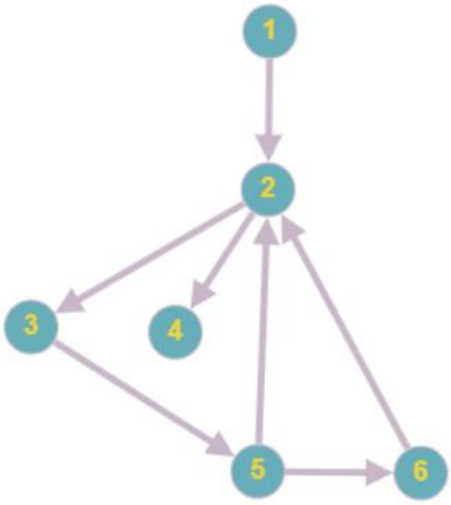
```
[TestMethod]
public async Task StudentServiceHome_LoopCoverageTest()
{
    _userManagerMock.Setup(m => m.GetUserAsync(It.IsAny<ClaimsPrincipal>())).ReturnsAsync(teacher);
    var exams = new List<Exam>
    {
        new Exam { Course = new Course { TeacherID = teacher.Id }, Time = DateTime.Now.AddDays(1) },
        new Exam { Course = new Course { TeacherID = 9999 }, Time = DateTime.Now.AddDays(1) },
        new Exam { Course = new Course { TeacherID = teacher.Id }, Time = DateTime.Now.AddDays(3) },
        new Exam { Course = new Course { TeacherID = teacher.Id }, Time = DateTime.Now.AddDays(5) }
    };

    _contextMock.Setup(c => c.Exam.Include(It.IsAny<Expression<Func<Exam, object>>>()))
        .Returns(exams.AsQueryable());
    var result = await controller.StudentServiceHome() as ViewResult;
    Assert.IsNotNull(result);
    Assert.IsTrue(result.ViewData.ContainsKey("Exams"));
    var examsInViewData = result.ViewData["Exams"] as List<Exam>;
    Assert.IsNotNull(examsInViewData);
    Assert.AreEqual(3, examsInViewData.Count);
    Assert.IsTrue(examsInViewData.Contains(exams[0]));
    Assert.IsTrue(examsInViewData.Contains(exams[2]));
    Assert.IsTrue(examsInViewData.Contains(exams[3]));
    Assert.IsFalse(examsInViewData.Contains(exams[1]));
}
```

5. Path coverage

Dijagram toka modula



Programski tok modula



Selma Ljuhar 19177

Tabela

Broj puta	Put
1	1 - 2 - 4
2	1 - 2 - 3 - 5 - 2 - 4
3	1 - 2 - 3 - 5 - 6 - 2 - 4

Testovi za dobijene puteve:

```
[TestMethod]
public void StudentServiceHome_Path1 ()
{
   _userManagerMock.Setup(m => m.GetUserAsync(It.IsAny<ClaimsPrincipal>())).ReturnsAsync(teacher);
   var exams = new List<Exam> { };
   _contextMock.Setup(c => c.Exam.Include(It.IsAny<Expression<Func<Exam, object>>>()))
      .Returns(exams.AsQueryable());
   var result = await controller.StudentServiceHome() as ViewResult;
   Assert.IsNotNull(result);
   Assert.IsTrue(result.ViewData.ContainsKey("Exams"));
   var examsInViewData = result.ViewData["Exams"] as List<Exam>;
   Assert.IsNotNull(examsInViewData);
   Assert.AreEqual(0, examsInViewData.Count);
}
```

```
[TestMethod]
public void StudentServiceHome_Path2 ()
{
   _userManagerMock.Setup(m => m.GetUserAsync(It.IsAny<ClaimsPrincipal>())).ReturnsAsync(teacher);
   var exams = new List<Exam>
   { new Exam { Course = new Course { TeacherID = 9999 }, Time = DateTime.Now.AddDays(1) }};
   _contextMock.Setup(c => c.Exam.Include(It.IsAny<Expression<Func<Exam, object>>>()))
      .Returns(exams.AsQueryable());
   var result = await controller.StudentServiceHome() as ViewResult;
   Assert.IsNotNull(result);
   Assert.IsTrue(result.ViewData.ContainsKey("Exams"));
   var examsInViewData = result.ViewData["Exams"] as List<Exam>;
   Assert.IsNotNull(examsInViewData);
   Assert.AreEqual(0, examsInViewData.Count);
}
```

Selma Ljuhar 19177

```
[TestMethod]
public void StudentServiceHome_Path3 ()
{
   _userManagerMock.Setup(m => m.GetUserAsync(It.IsAny<ClaimsPrincipal>())).ReturnsAsync(teacher);
   var exams = new List<Exam> { new Exam { Course = new Course { TeacherID = teacher.Id }, Time =
DateTime.Now.AddDays(1) }};
   _contextMock.Setup(c => c.Exam.Include(It.IsAny<Expression<Func<Exam, object>>>()))
      .Returns(exams.AsQueryable());
   var result = await controller.StudentServiceHome() as ViewResult;
   Assert.IsNotNull(result);
   Assert.IsTrue(result.ViewData.ContainsKey("Exams"));
   var examsInViewData = result.ViewData["Exams"] as List<Exam>;
   Assert.IsNotNull(examsInViewData);
   Assert.AreEqual(1, examsInViewData.Count);
   Assert.AreEqual(exams[0], examsInViewData[0]);
}
```