Изпит – Unit тестове с InMemory база данни и Mock обекти

Тема на изпита:

Тестване на ASP.NET Core MVC приложение „SchoolCourses“, което използва Entity Framework Core за работа с база данни, чрез:
• InMemory база данни (Microsoft.EntityFrameworkCore.InMemory)
• Mock обекти (Moq) за външни зависимости

Описание на готовия проект:

Проектът съдържа ASP.NET Core MVC приложение със следните основни елементи:
• Модели (Entities): Student, Course, Enrollment
• DbContext: SchoolDbContext
• Сервизни класове: StudentService, CourseService, EnrollmentService
• Интерфейс за изпращане на имейли: IEmailSender, имплементиран от EmailSender

Задача 1. Създаване на тестов проект (5 т.)

1. Създайте нов NUnit Test Project с име “SchoolCourses.Tests”.
2. Добавете референция към основния проект “SchoolCourses”.
3. Инсталирайте следните NuGet пакети в тестовия проект:
   • Microsoft.EntityFrameworkCore.InMemory
   • Moq
   • NUnit3TestAdapter
   • Microsoft.NET.Test.Sdk

Задача 2. Unit тестове за StudentService (10 т.)

Използвайки InMemory база данни, създайте тестове за класа StudentService:

1. Тест, който проверява, че методът GetAverageGradeAsync връща null, когато ученикът няма оценки.
2. Тест, който проверява, че GetAverageGradeAsync връща коректната средна оценка, когато има поне две оценки.
3. Тест, който проверява, че GetAllAsync връща учениците сортирани по фамилия, а при еднаква фамилия – по собствено име.

Задача 3. Unit тестове за EnrollmentService с InMemory и Moq (15 т.)

Използвайки InMemory база данни и mock на IEmailSender (Moq), създайте тестове за класа EnrollmentService:

1. Тест, който проверява, че EnrollStudentAsync връща false, когато ученикът или курсът не съществуват.
2. Тест, който проверява, че EnrollStudentAsync не позволява записване на един и същи ученик в един и същи курс втори път.
3. Тест, който проверява, че при успешно записване:
   • се създава нов запис в таблицата Enrollments в InMemory базата;
   • се извиква методът IEmailSender.SendAsync точно веднъж.
4. Тестове за SetGradeAsync:
   • връща false, когато оценката е извън диапазона [2.00; 6.00];
   • връща true и записва оценката, когато входните данни са валидни и записа съществува.

Задача 4. Допълнителни тестове (10 т.)

Добавете поне два допълнителни теста по ваш избор за някой от следните случаи:

• Негативни сценарии (невалидни Id, липсващи записи, празни колекции).
• Тест за CourseService.GetStudentsInCourseAsync – проверка за правилен брой и ред на върнатите ученици.
• Други гранични случаи, които смятате за важни.

Оценяване:

Максимален брой точки: 40
30–40 т. – Отличен (6)
24–29 т. – Много добър (5)
18–23 т. – Добър (4)
12–17 т. – Среден (3)
0–11 т. – Слаб (2)

Указания:

• Всеки тест трябва да има ясно име и да следва структура Arrange–Act–Assert.
• Използвайте отделна InMemory база данни за всеки тест (различно databaseName).

• След приключване на всяка задача, стартирайте всички тестове и се уверете, че минават успешно.