We have a web application for students so they can use different features. We want to introduce a new feature which allows users to create and join study groups for different subjects. This feature has an API with the code you can find below. Behind the scene, the system stores data in the repository, which is implemented as a MS SQL database. Although Subjects and Users are already existing in our data model, we want to introduce a new entity for StudyGroups, so you can also check below the code introduced for it.

The task

We need you to do the following as part of this task:

1. Write a list of different test cases to check this feature and the integrity of new entity StudyGroups according to the acceptance criteria:
    a. Describe some high level steps and expectations (you can make assumptions of how the app works - just explain it)
    b. Highlight what are the inputs you will be using on each test case
    c. Define testing level of this test case: unit testing, component testing or e2e testing (manual) - considering that:
        i. We have a unit test framework in TestApp using Nunit framework
        ii. We have a component test framework in our TestAppAPI using Nunit framework
        iii. We don't have any automation to test the UI, so manual testing will be required on e2e level
    d. Consider if you want to add all test cases to regression or not
2. Write the code for all automated tests you described on the different frameworks
3. Write a SQL query that will return "all the StudyGroups which have at least an user with 'name' starting on 'M' sorted by 'creation date'" like "Miguel" or "Manuel".
4. Organize the outcome of your work best for readers and reviewers - use MS Word/Excel, GitLab or GitHub.

Acceptance criteria

1. Users are able to create only one Study Group for a single Subject
    a. Users can provide a name for the Study Group with size between 5-30 characters
    b. The only valid Subjects are: Math, Chemistry, Physics
    c. We want to record when Study Groups were created
2. Users can join Study Groups for different Subjects
3. Users can check the list of all existing Study Groups
    a. Users can also filter Study Groups by a given Subject
    b. Users can sort to see most recently created Study Groups or oldest ones
4. Users can leave Study Groups they joined

API Controller class

```
using Microsoft.AspNetCore.Mvc;

namespace TestAppAPI
{
    public class StudyGroupController
    {
        private readonly IStudyGroupRepository _studyGroupRepository;

        public StudyGroupController(IStudyGroupRepository studyGroupRepository)
        {
            _studyGroupRepository = studyGroupRepository;
        }

        public async Task<IActionResult> CreateStudyGroup(StudyGroup studyGroup)
        {
            await _studyGroupRepository.CreateStudyGroup(studyGroup);
            return new OkResult();
        }

        public async Task<IActionResult> GetStudyGroups()
        {
            var studyGroups = await _studyGroupRepository.GetStudyGroups();
            return new OkObjectResult(studyGroups);
        }

        public async Task<IActionResult> SearchStudyGroups(string subject)
        {
            var studyGroups = await _studyGroupRepository.SearchStudyGroups(subject);
            return new OkObjectResult(studyGroups);
        }

        public async Task<IActionResult> JoinStudyGroup(int studyGroupId, int userId)
        {
            await _studyGroupRepository.JoinStudyGroup(studyGroupId, userId);
            return new OkResult();
        }

        public async Task<IActionResult> LeaveStudyGroup(int studyGroupId, int userId)
        {
            await _studyGroupRepository.LeaveStudyGroup(studyGroupId, userId);
            return new OkResult();
```

```
        }
    }
}
```

StudyGroup class

```
using System;
using System.Collections.Generic;

namespace TestApp
{
    public class StudyGroup
    {
        public StudyGroup(int studyGroupId, string name, Subject subject, DateTime create
        {
            StudyGroupId = studyGroupId;
            Name = name;
            Subject = subject;
            CreateDate = createDate;
            Users = users;
        }
        //Some logic will be missing to validate values according to acceptance criteria,
        public int StudyGroupId { get; }

        public string Name { get; }

        public Subject Subject { get; }

        public DateTime CreateDate { get; }

        public List<User> Users { get; private set; }

        public void AddUser(User user)
        {
            Users.Add(user);
        }

        public void RemoveUser(User user)
        {
            Users.Remove(user);
        }
    }

    public enum Subject
    {
```

```
        Math,
        Chemistry,
        Physics
    }
}
```