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:
   1. Describe some high level steps and expectations (you can make assumptions of how the app works - just explain it)
   2. Highlight what are the inputs you will be using on each test case
   3. Define testing level of this test case: unit testing, component testing or e2e testing (manual) - considering that:
      1. We have a unit test framework in TestApp using Nunit framework
      2. We have a component test framework in our TestAppAPI using Nunit framework
      3. We don't have any automation to test the UI, so manual testing will be required on e2e level
   4. 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
   1. Users can provide a name for the Study Group with size between 5-30 characters
   2. The only valid Subjects are: Math, Chemistry, Physics
   3. 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
   1. Users can also filter Study Groups by a given Subject
   2. 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 createDate, List<User> users)          {              StudyGroupId = studyGroupId;              Name = name;              Subject = subject;              CreateDate = createDate;              Users = users;          }          //Some logic will be missing to validate values according to acceptance criteria, but imagine it is existing or do it yourself          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      }  } |

Resolução

Para começar a resolver este exercício, siga estes passos:

**1-Análise dos Requisitos: Entenda completamente os requisitos e critérios de aceitação para a funcionalidade de grupos de estudo.**

*R- A análise dos requisitos para a funcionalidade de grupos de estudo implica compreender que os usuários podem criar e se juntar a grupos por disciplinas específicas, sendo restritos a criar apenas um grupo por matéria. É necessário que os grupos tenham nomes com um limite de caracteres e que as disciplinas válidas sejam Matemática, Química e Física. Além disso, a funcionalidade deve permitir aos usuários visualizar, filtrar e ordenar os grupos existentes, bem como sair de grupos dos quais fazem parte, mantendo o registro de quando cada grupo foi criado.*

**2-Identificação dos Casos de Teste: Baseado nos critérios, identifique casos de teste abrangendo criação, adesão, visualização e saída de grupos de estudo, além da integridade da entidade StudyGroups.**

*R-*

*Criação de Grupo: Testar a criação de um grupo por assunto por usuário, validação do nome (5-30 caracteres), e restrição dos assuntos válidos (Matemática, Química, Física).*

*Adesão a Grupo: Verificar a possibilidade de um usuário juntar-se a múltiplos grupos de diferentes assuntos e restrição de não aderir ao mesmo grupo mais de uma vez.*

*Visualização de Grupos: Testar a funcionalidade de listar todos os grupos disponíveis, incluindo a filtragem por assunto e a ordenação por data de criação.*

*Saída de Grupos: Avaliar o processo de um usuário sair de um grupo do qual é membro, garantindo que a ação seja refletida corretamente nos dados do grupo.*

*Integridade da Entidade StudyGroups: Verificar a consistência e integridade dos dados relacionados aos grupos de estudo, como o registro correto de criação de grupos e a atualização correta da lista de membros ao entrar ou sair de grupos.*

*Desenvolvimento dos Testes Automatizados: Escreva o código para testes unitários e de componente utilizando o framework NUnit para as lógicas de negócio e integração da API.*

*Planejamento dos Testes E2E Manuais: Defina cenários para testes manuais da interface do usuário, cobrindo todas as interações possíveis com a funcionalidade de grupos de estudo.*

*Elaboração da Query SQL: Desenvolva a query SQL que retorna os grupos de estudo conforme os critérios especificados.*

*Documentação e Organização: Documente todos os casos de teste, o código desenvolvido e os resultados obtidos de maneira clara e organizada, utilizando ferramentas como MS Word/Excel e plataformas como GitLab ou GitHub para compartilhamento.*

*Iniciar por uma compreensão profunda dos requisitos e critérios de aceitação é crucial para garantir que todos os aspectos da funcionalidade sejam devidamente testados e validados.*

**3- Desenvolvimento dos Testes Automatizados: Escreva o código para testes unitários e de componente utilizando o framework NUnit para as lógicas de negócio e integração da API.**

*R-*

*Para desenvolver testes automatizados com o NUnit, você criaria classes de teste específicas para cada componente ou unidade da funcionalidade de grupos de estudo. Para testes unitários, focaria em lógicas individuais dentro da classe StudyGroup, como a adição e remoção de usuários, validação do nome do grupo e restrição de assunto. Nos testes de componente, abordaria a integração da classe StudyGroupController com o repositório, testando a criação de grupos, adesão, busca e saída de grupos de estudo. Utilizaria atributos do NUnit como [TestFixture] para a classe de teste, [Test] para métodos de teste, e assertivas para validar os resultados esperados.*

**4-Planejamento dos Testes E2E Manuais: Defina cenários para testes manuais da interface do usuário, cobrindo todas as interações possíveis com a funcionalidade de grupos de estudo.**

*R - Para planejar testes E2E manuais da interface do usuário para a funcionalidade de grupos de estudo, considere cenários como: criação de um grupo de estudo, validando a restrição de um grupo por matéria por usuário; adesão a grupos existentes e verificação da limitação de adesão; visualização e filtragem de grupos por matéria e ordenação por data de criação; e por fim, a saída de um grupo de estudo, verificando a atualização correta do status do usuário e do grupo.*

**5-Elaboração da Query SQL: Desenvolva a query SQL que retorna os grupos de estudo conforme os critérios especificados.**

*Para retornar os grupos de estudo que tenham pelo menos um usuário com o nome começando com "M", ordenados pela data de criação, você pode usar a seguinte query SQL:*

*SELECT sg.\* FROM StudyGroups sg*

*JOIN Users u ON sg.StudyGroupId = u.StudyGroupId*

*WHERE u.Name LIKE 'M%'*

*GROUP BY sg.StudyGroupId*

*ORDER BY sg.CreateDate;*

**6- Documentação e Organização: Documente todos os casos de teste, o código desenvolvido e os resultados obtidos de maneira clara e organizada, utilizando ferramentas como MS Word/Excel e plataformas como GitLab ou GitHub para compartilhamento.**

*R –*

*Para documentar e organizar os casos de teste, o código desenvolvido e os resultados obtidos, siga estes passos:*

*Utilize MS Word ou Excel para criar documentos detalhados dos casos de teste, incluindo descrição, passos, entradas, resultados esperados e observados.*

*Armazene e compartilhe o código no GitHub ou GitLab, criando repositórios para o código dos testes automatizados e scripts SQL. Inclua um README com instruções de como executar os testes e qualquer dependência necessária.*

*Documente os resultados dos testes, incluindo screenshots para testes manuais e logs ou relatórios de execução dos testes automatizados.*

*Mantenha a documentação atualizada, revisando-a regularmente para refletir quaisquer mudanças nos requisitos, no código ou nos resultados dos testes.*

*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*

* 1. *Consider if you want to add all test cases to regression or not*

*Para atender à tarefa solicitada, os casos de teste para a funcionalidade de grupos de estudo devem incluir:*

*Criação de Grupo de Estudo:*

*Passos: Tentativa de criação de grupo com nome válido, seleção de matéria e verificação da criação.*

*Entradas: Nome do grupo (5-30 caracteres), matéria (Matemática, Química, Física).*

*Nível: Unitário para lógica de validação, Componente para interação com o repositório, E2E para processo na UI.*

*Regressão: Sim.*

*Adesão a Grupo Existente:*

*Passos: Usuário tenta se juntar a um grupo existente.*

*Entradas: ID do usuário, ID do grupo de estudo.*

*Nível: Componente, E2E manual para verificar a UI.*

*Regressão: Sim.*

*Visualização e Filtragem de Grupos:*

*Passos: Usuário visualiza todos os grupos e aplica filtro por matéria.*

*Entradas: Matéria para filtro.*

*Nível: E2E manual.*

*Regressão: Opcional.*

*Saída de Grupo de Estudo:*

*Passos: Usuário sai de um grupo de estudo.*

*Entradas: ID do usuário, ID do grupo de estudo.*

*Nível: Componente, E2E manual.*

*Regressão: Sim.*

*Estes casos de teste cobrem os aspectos fundamentais da funcionalidade de grupos de estudo, desde a criação até a saída dos grupos, passando pela adesão e visualização/filtragem.*

*Decidir se um caso de teste deve ir para a suíte de regressão depende de vários fatores, como a criticidade da funcionalidade que ele cobre, a frequência com que essa área do código é alterada e o esforço necessário para manter o teste. Em geral, casos de teste que cobrem funcionalidades centrais e críticas do sistema, como a criação de grupos de estudo e a adesão a eles, devem fazer parte da suíte de regressão para garantir que qualquer modificação no código não quebre essas funcionalidades essenciais.*

**2. Write the code for all automated tests you described on the different frameworks**

*Para escrever os testes automatizados descritos, você usaria o NUnit para criar testes unitários e de componentes verificando a lógica de negócios da aplicação e a integração da API, respectivamente. Testes E2E manuais seriam documentados separadamente, detalhando os passos para interação direta com a interface do usuário.*

*Baseando-me no enunciado e nas respostas fornecidas até agora, minha proposta de código para testes automatizados usando NUnit envolveria:*

*Testes Unitários para a Classe StudyGroup: Verificar a adição e remoção de usuários, além da validação do nome e assunto do grupo de estudo.*

*Testes de Componentes para StudyGroupController: Testar a integração entre os métodos da API e o repositório, como a criação de grupos, adesão, visualização e saída de grupos.*

*Cada teste automatizado focaria em um aspecto específico da funcionalidade, utilizando asserções para validar os resultados esperados. Recomendo consultar a documentação do NUnit para exemplos específicos de como implementar esses testes no contexto da sua aplicação.*

***ESSA RESPOSTA AINDA CONTINUA, PRECISA DE MAIS IMPLEMENTAÇÃO DE TESTES***

**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".**

*R-*

*SELECT DISTINCT sg.\* FROM StudyGroups sg*

*JOIN Users u ON sg.StudyGroupId = u.StudyGroupId*

*WHERE u.Name LIKE 'M%'*

*ORDER BY sg.CreateDate;*

**4.Organize the outcome of your work best for readers and reviewers - use MS Word/Excel, GitLab or GitHub**

*https://github.com/herbertfilhik/StudyGroupsManager/blob/main/README.md*