Project work 1

Breif

The task this week is to take on an issue from the project board and use the techniques learned so far such as good programming practices, testing and code reviews.

Issue Task

The Issue task I have worked on has the folowing description:

As an UNDAC Team Support and Logistics Manager, I want to request the removal of access for uses when they leave so that effective security is maintained

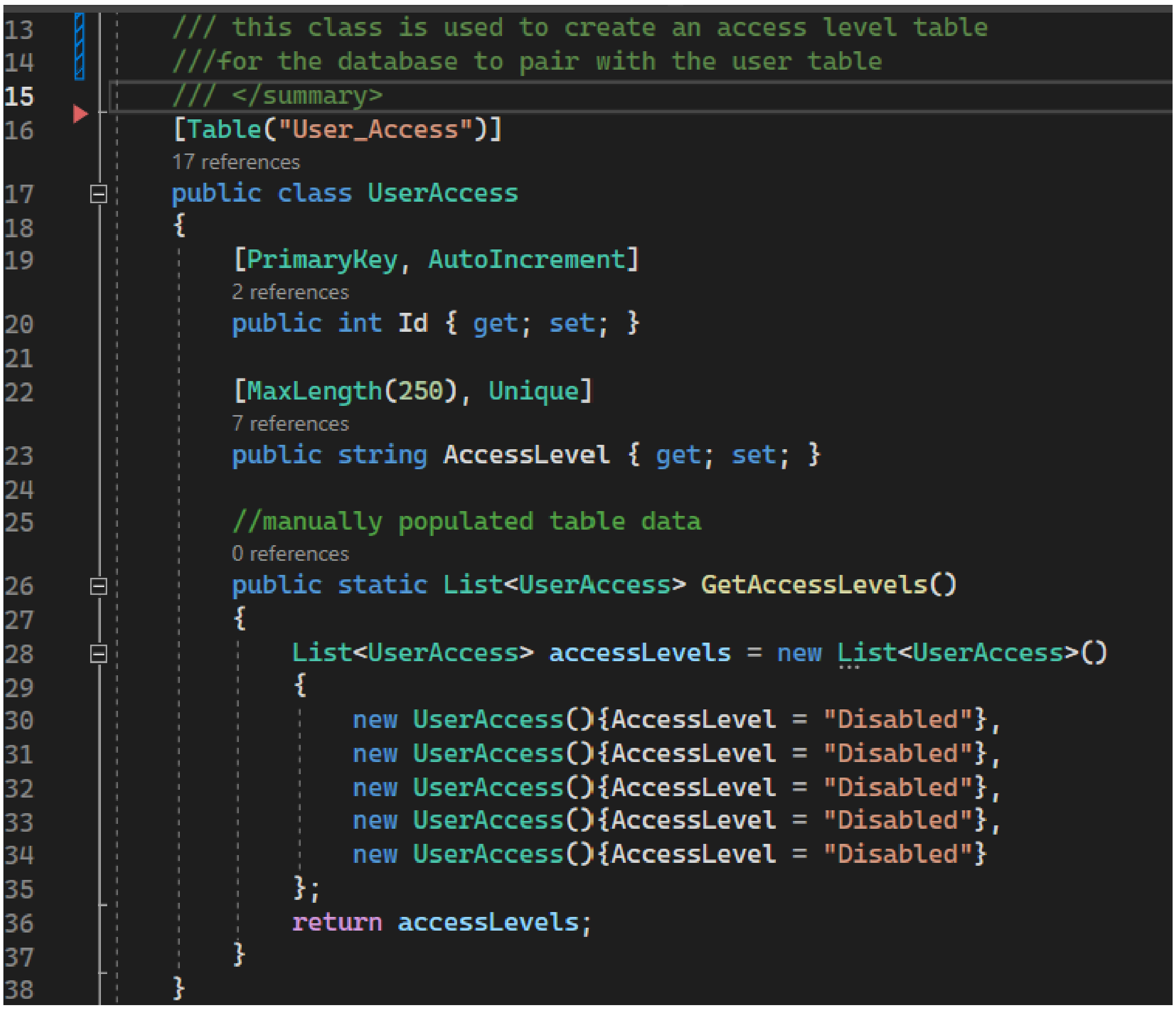
End user goal: To prevent access to mission system once it is no longer needed

End business goal: To ensure the data security of the mission

Acceptance criteria:

- ***Requests for removal of access are automatically approved***
- ***Records are not delete when access is removed - they are set to a status of 'disabled'***

To achieve this I worked on creating a table as shown below:



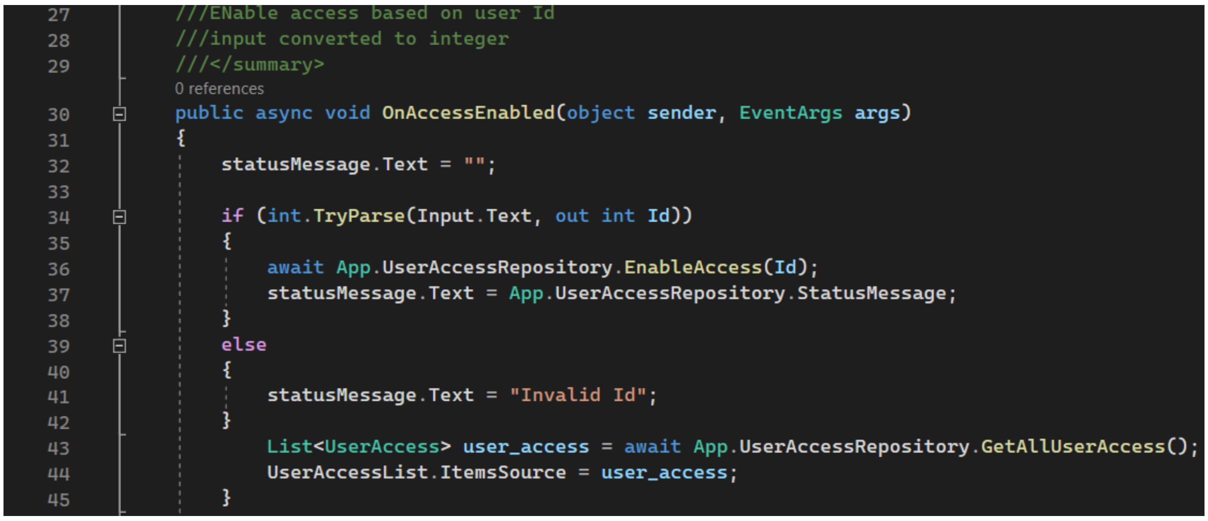
```csharp
    /// this class is used to create an access level table
    ///for the database to pair with the user table
    /// </summary>
    [Table("User_Access")]
    public class UserAccess
    {
        [PrimaryKey, AutoIncrement]
        public int Id { get; set; }

        [MaxLength(250), Unique]
        public string AccessLevel { get; set; }

        //manually populated table data
        public static List<UserAccess> GetAccessLevels()
        {
            List<UserAccess> accessLevels = new List<UserAccess>()
            {
                new UserAccess(){AccessLevel = "Disabled"},
                new UserAccess(){AccessLevel = "Disabled"},
                new UserAccess(){AccessLevel = "Disabled"},
                new UserAccess(){AccessLevel = "Disabled"},
                new UserAccess(){AccessLevel = "Disabled"}
            };
            return accessLevels;
        }
    }
```

Fig.1 User Access Table

The table is to be paired with a user table in the database so that based on user ID we can assign their access levels to be either disabled or enabled. As there is currently no User table created yet I have populated the table manualy with dummy data to prototype the functionality of this part of the system.

As the code is self explanetory I have minimised use of comments and reserved them for documentation comments.

I then created a class that will take user input on the user access page to disable/enable access.



```csharp
    ///ENable access based on user Id
    ///input converted to integer
    ///</summary>
    public async void OnAccessEnabled(object sender, EventArgs args)
    {
        statusMessage.Text = "";

        if (int.TryParse(Input.Text, out int Id))
        {
            await App.UserAccessRepository.EnableAccess(Id);
            statusMessage.Text = App.UserAccessRepository.StatusMessage;
        }
        else
        {
            statusMessage.Text = "Invalid Id";
        }
            List<UserAccess> user_access = await App.UserAccessRepository.GetAllUserAccess();
            UserAccessList.ItemsSource = user_access;
    }
```

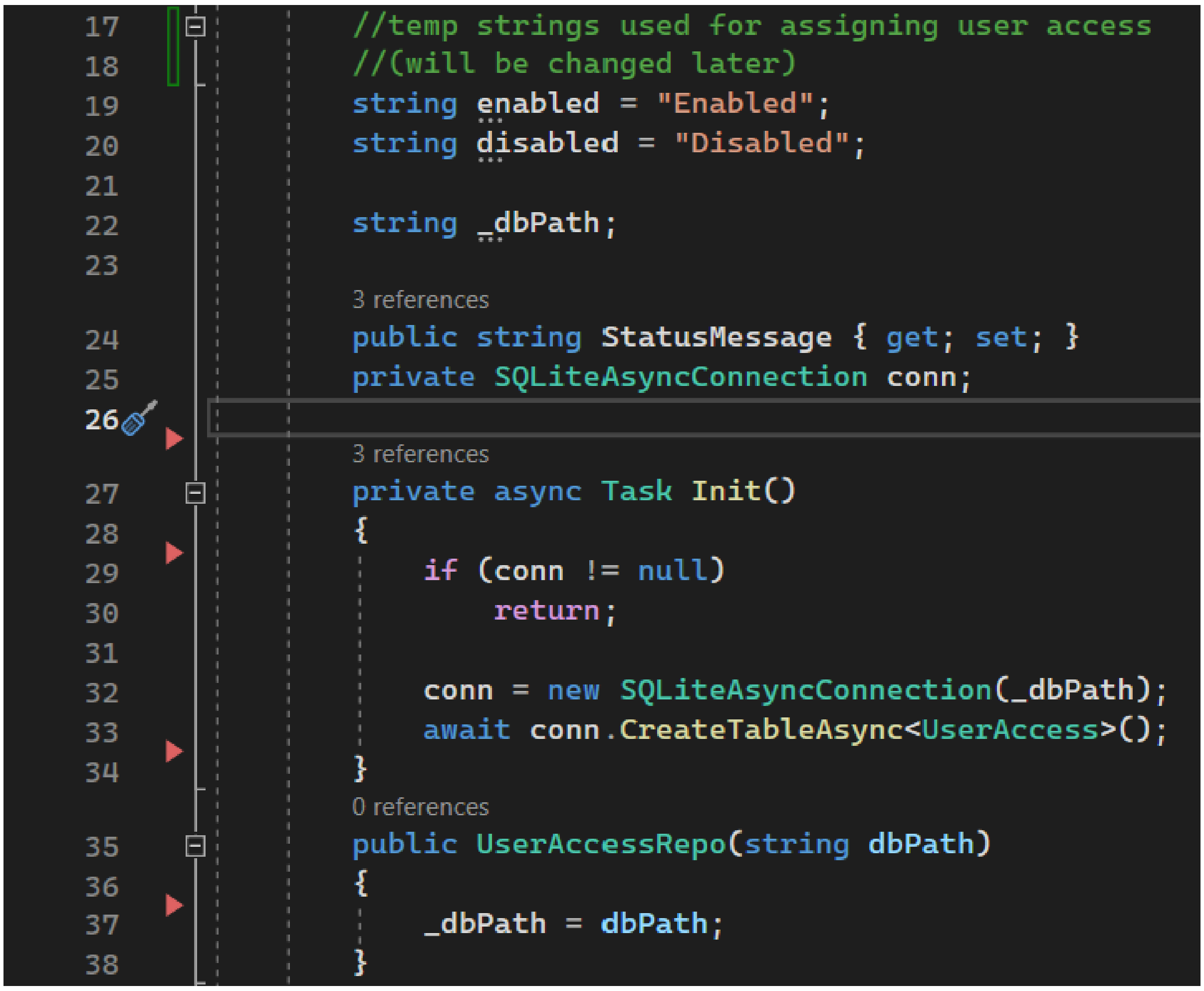
Fig.2 AccessPage enabling access

When a User inputs an ID number they can select the enable access button which will call the method shown in Fig.2 this method will then convert user input to an integer value or otherwise inform the user that the input field is invalid. If the correct input is entered then the method will call the "EnableAccess()" method from the "UserAccessRepo" class passing in the input value.

Regardless of correct input when this method is called there will be a refreshed list display from the User_Access table which is achieved by calling the "GetAllUserAccess()" method from the "UserAccessRepo" class.

The second function of this class is the disable button which is the exact same code however the only difference being it calls the "DisableAccess()" method instead of "EnableAccess()".

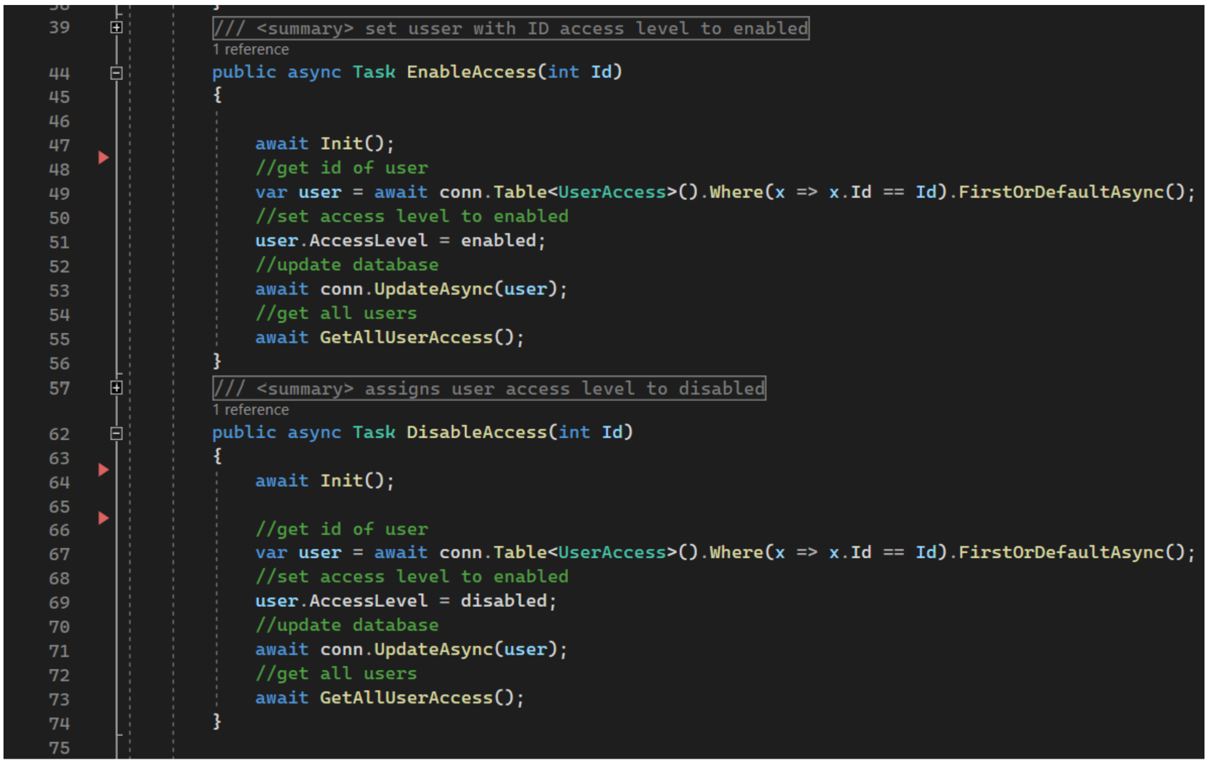
The "UserAccesssRepo" handles the data manipulation of the User_Access table and is used by the inputs from the user on the "UserAccessPage" that was just discussed prior.



```
17          //temp strings used for assigning user access
18          //(will be changed later)
19          string enabled = "Enabled";
20          string disabled = "Disabled";
21
22          string _dbPath;
23
            3 references
24          public string StatusMessage { get; set; }
25          private SQLiteAsyncConnection conn;
26
            3 references
27          private async Task Init()
28          {
29              if (conn != null)
30                  return;
31
32              conn = new SQLiteAsyncConnection(_dbPath);
33              await conn.CreateTableAsync<UserAccess>();
34          }
            0 references
35          public UserAccessRepo(string dbPath)
36          {
37              _dbPath = dbPath;
38          }
```

Fig.3 User Access Repo Prep

Fig.3 shows the basic prep for the class such as two strings for handling setting access levels (lines 19-20) and the database string(line 22) which is used to store the database path. Line 24 is used for the display messages to the user and Line 25 is used to get our connection to the database. the rest of the code in Fig.3 is ensuring there is a database connection and creating one if there isnt already.



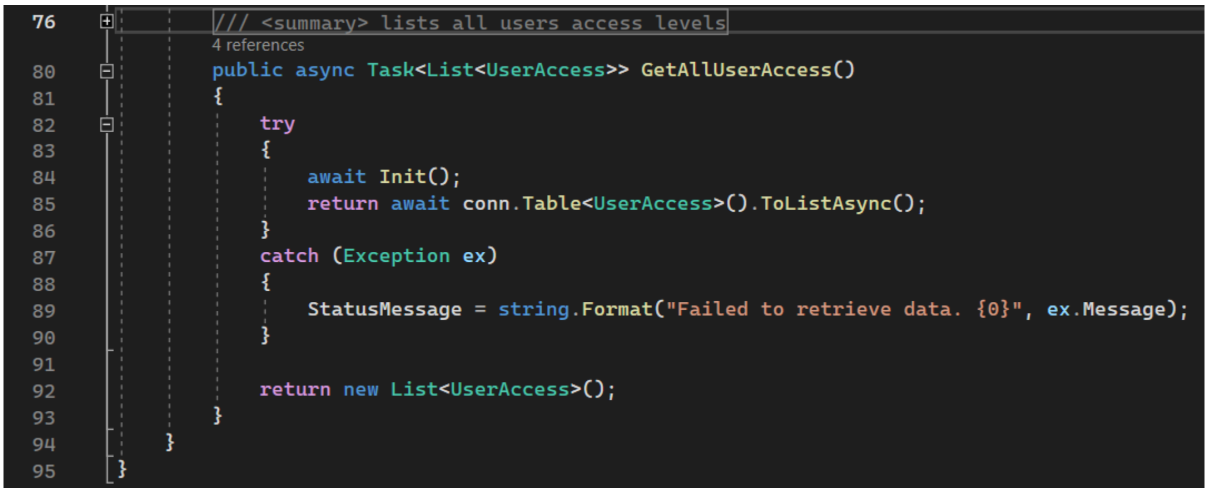
```
39      /// <summary> set usser with ID access level to enabled
        1 reference
44      public async Task EnableAccess(int Id)
45      {
46
47          await Init();
48          //get id of user
49          var user = await conn.Table<UserAccess>().Where(x => x.Id == Id).FirstOrDefaultAsync();
50          //set access level to enabled
51          user.AccessLevel = enabled;
52          //update database
53          await conn.UpdateAsync(user);
54          //get all users
55          await GetAllUserAccess();
56      }
57      /// <summary> assigns user access level to disabled
        1 reference
62      public async Task DisableAccess(int Id)
63      {
64          await Init();
65
66          //get id of user
67          var user = await conn.Table<UserAccess>().Where(x => x.Id == Id).FirstOrDefaultAsync();
68          //set access level to enabled
69          user.AccessLevel = disabled;
70          //update database
71          await conn.UpdateAsync(user);
72          //get all users
73          await GetAllUserAccess();
74      }
75
```

Fig.4 User Access Repo Enable/Disable Methods

Lines 44-55 of Fig.4 show the Method used to update the table row (based on the user Id)with the new access value. here in line 51 we simply use the prepared string shown in line 19 of Fig.3 to pass in the new enable value. Once these changes have been made the "GetAllUserAccess()"method will be called.

The method for dissabling access is identical with the only difference of using the prepared string for disabling access.

The "GetAllUserAccess()" method is used for displaying the table to the users on screen.



```
        /// <summary> lists all users access levels
        4 references
        public async Task<List<UserAccess>> GetAllUserAccess()
        {
            try
            {
                await Init();
                return await conn.Table<UserAccess>().ToListAsync();
            }
            catch (Exception ex)
            {
                StatusMessage = string.Format("Failed to retrieve data. {0}", ex.Message);
            }

            return new List<UserAccess>();
        }
    }
}
```

Fig.5 User Access Repo Display List

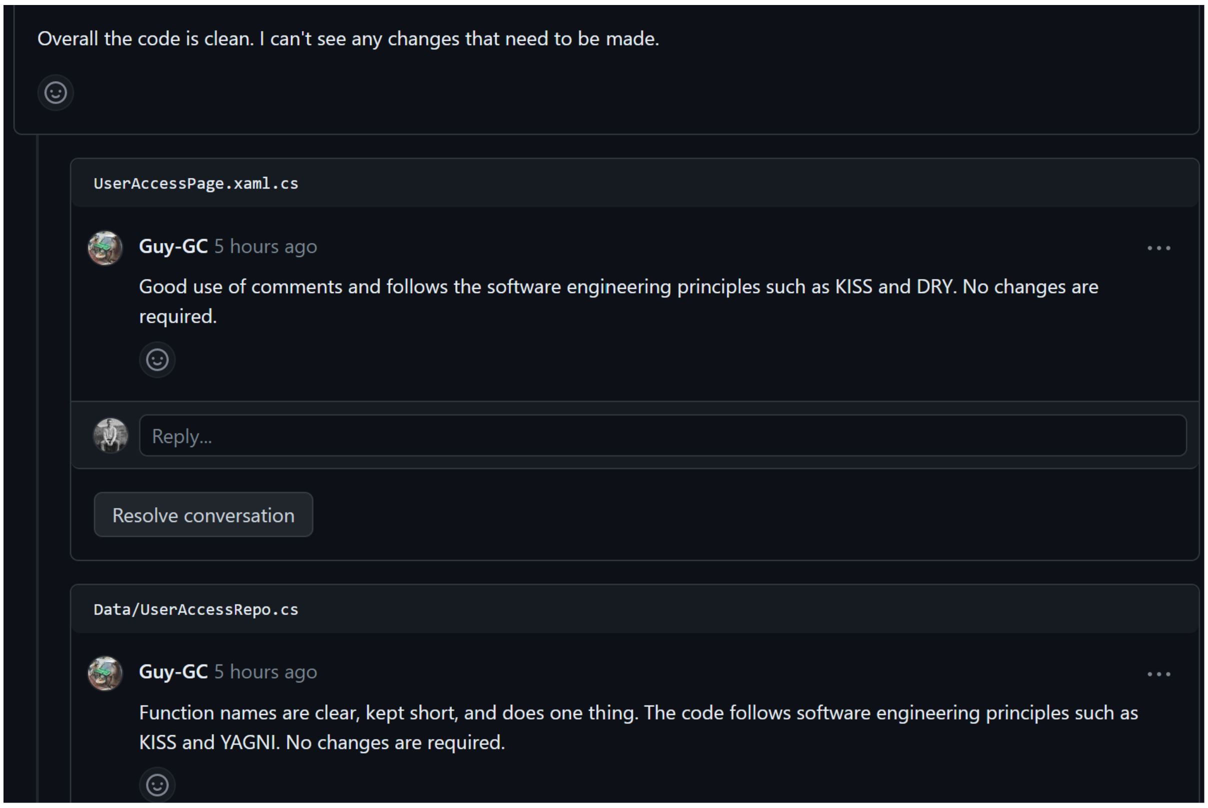
This method returns the data from the table into a list that is used to populate the collections view of the "UserAccessPage.xml" file so that users of the system can see the changes made aswell as view the list of access levels assigned to each person.

Testing

The Testing framework I had tried to use was Xunit testing which Is the method team members decided to try using, however team members reported that this was causing issues in the repository and I was also having issues setting it up so the team members who are having issues and myself have decided to postpone testing to the in-person meeting during week 9 and work on setting this up together as a team so we can have a safer more uniform set up as although it is benifitial to do testing earlier than later we felt its not crucial to our workflow for these tasks.

Code Reviews

Feedback On My Code



Overall the code is clean. I can't see any changes that need to be made.

UserAccessPage.xaml.cs

Guy-GC 5 hours ago

Good use of comments and follows the software engineering principles such as KISS and DRY. No changes are required.

Data/UserAccessRepo.cs

Guy-GC 5 hours ago

Function names are clear, kept short, and does one thing. The code follows software engineering principles such as KISS and YAGNI. No changes are required.

Fig.6 Code Review on my pull request

The Code Review I have recieved was possitive and didnt prompt for any fixes or improvements.

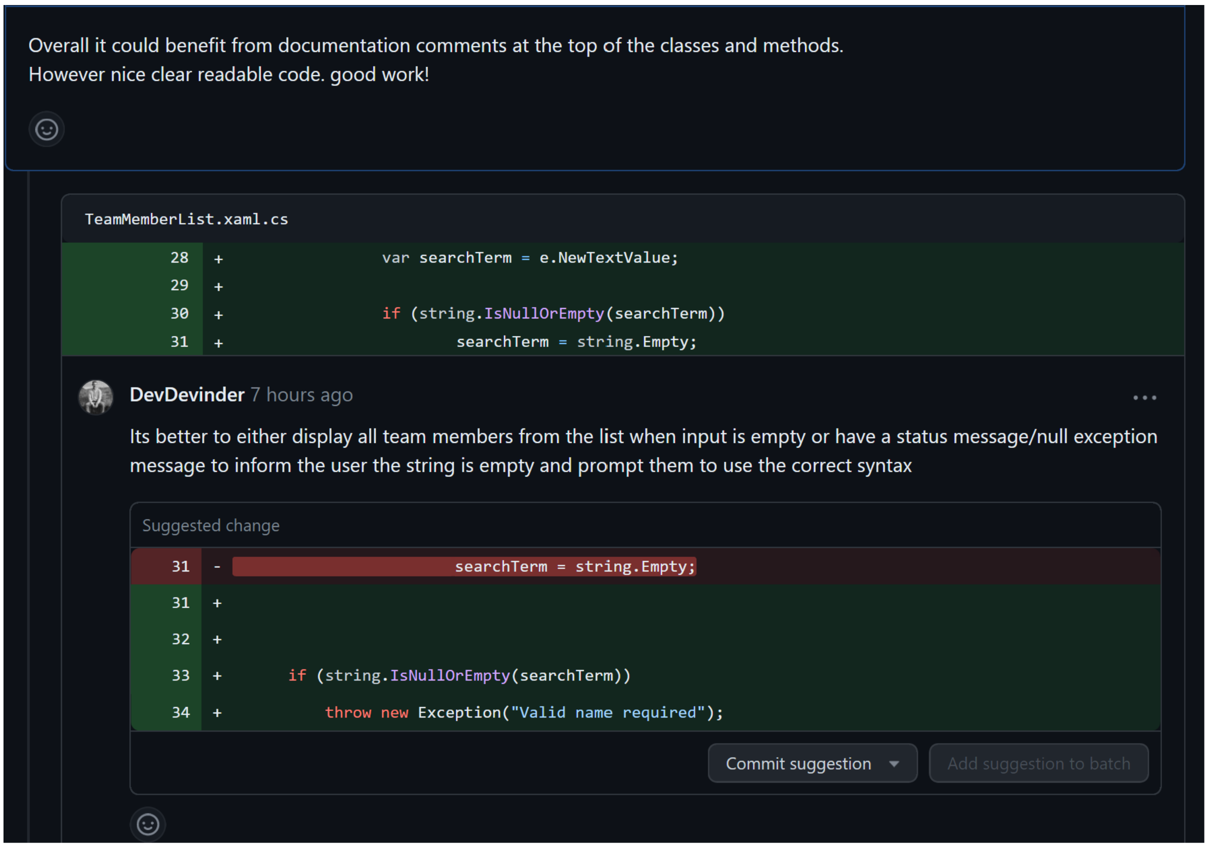
Feedback On My Own Code

As my Code review didnt have any comments regarding improvements I felt neccessary to add a more critical review of my code.

I feel Fig.4 shows extreemly uneccessary comments such as line 54 and in Fig.3 lines 19-20 I could remove the comments entirely and have better names for the temp variables.

I also feel that my code could benifit from more error handling with Try-Catch blocks to improve maintainability and internal quality.

Feedback I Have Given To Team Member



Overall it could benefit from documentation comments at the top of the classes and methods.
However nice clear readable code. good work!

TeamMemberList.xaml.cs

```
28 +                var searchTerm = e.NewTextValue;
29 +
30 +                if (string.IsNullOrEmpty(searchTerm))
31 +                    searchTerm = string.Empty;
```

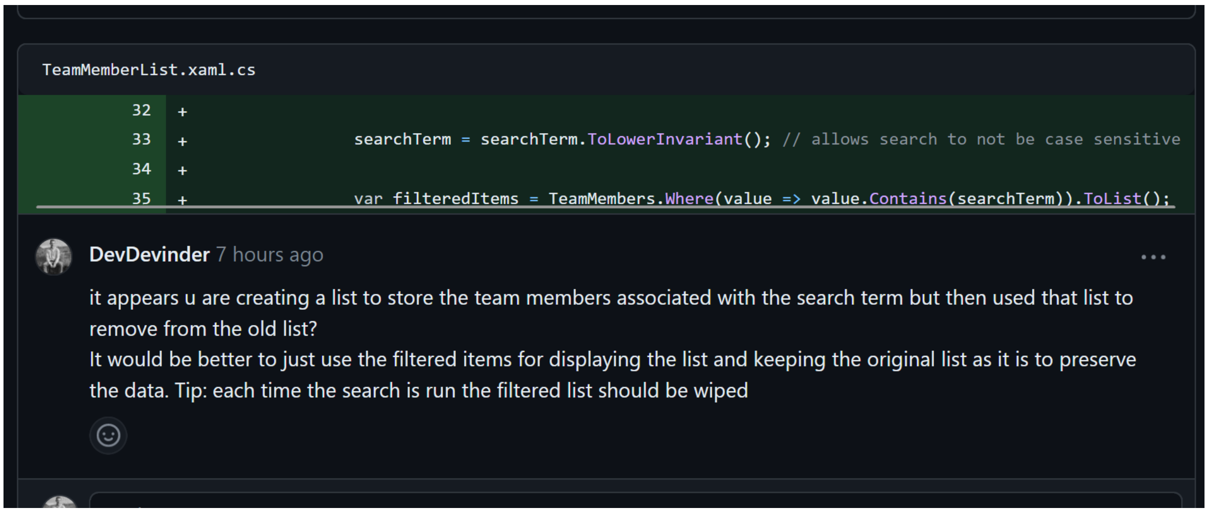
DevDevinder 7 hours ago

Its better to either display all team members from the list when input is empty or have a status message/null exception message to inform the user the string is empty and prompt them to use the correct syntax

Suggested change

```
31 -                    searchTerm = string.Empty;
31 +
32 +
33 +      if (string.IsNullOrEmpty(searchTerm))
34 +          throw new Exception("Valid name required");
```

Fig.7 Code Review Of Team Member



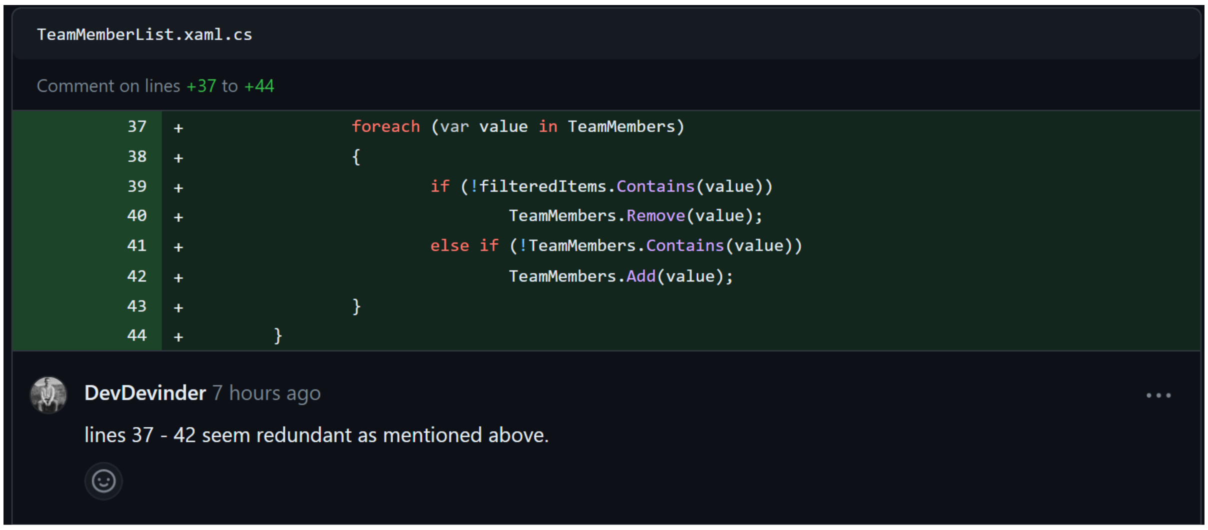
TeamMemberList.xaml.cs

```
32 +
33 +                searchTerm = searchTerm.ToLowerInvariant(); // allows search to not be case sensitive
34 +
35 +                var filteredItems = TeamMembers.Where(value => value.Contains(searchTerm)).ToList();
```

DevDevinder 7 hours ago

it appears u are creating a list to store the team members associated with the search term but then used that list to remove from the old list?
It would be better to just use the filtered items for displaying the list and keeping the original list as it is to preserve the data. Tip: each time the search is run the filtered list should be wiped

Fig.8 Code Review Of Team Member



TeamMemberList.xaml.cs

Comment on lines +37 to +44

```
37 +                foreach (var value in TeamMembers)
38 +                {
39 +                    if (!filteredItems.Contains(value))
40 +                        TeamMembers.Remove(value);
41 +                    else if (!TeamMembers.Contains(value))
42 +                        TeamMembers.Add(value);
43 +                }
44 +            }
```

DevDevinder 7 hours ago

lines 37 - 42 seem redundant as mentioned above.

Fig.9 Code Review Of Team Member

When Reviewing my team members code I found some code seemed to be uneccessery, that may produce technical debt and is breaching the "KISS" principle such as noted both Fig.8 and Fig.9. (although I sense it was only breached because team member felt the complexity was neccessary here)

I also found that the internal quality could be improved as shown on Fig.7 in regards to exception handling when dealing with nulls, although Im aware I could have expressed that better and chosen a better suggestion such as using a try catch.

overall I can see the code is well written with KISS in mind(dispite my views mentioned above) and has followed the YAGNI principle. It is clear and readible to understand what is happening without comments however the code is lacking in documentation comments which could be helpfull more so as the project grows.

Reflection

Code Comparison

When comparing my code to others I noticed that others have more readible and concise code while mine tends to be more aiming for robustness which can add more complexity to simpler tasks. (however this is in general as I feel this weeks task my code is quite basic)

What I Have Learned

I have learned the importance of code reviews and the benifits of them as I feel more inclined to improve my code practices while writing them when I am aware somone is going to review it. (especially as it may result in having to rewrite it)

Another reason I found them helpful is that it helps to keep you right and you can learn from others opinions and code, regardless whether your reviewing theirs or they are reviewing yours.

I found that when working in a large team on a singular project there needs to be a strong level of communication and regular meetings to ensure that everyone is on the same page and to minimise issues when merging and delays during development.

Due to this I can see the benifits of having an agreed and upto date workflow, however reaching this point in the submissions have enforeced the realisation that creating a design documentation in the begining would have given the team higher guidance to ballence out where communication is lacking.

Another problem I faced is that when working in a team it can become problematic when changing the working environment such as adding new dependancies and packages as you need to ensure everyone is working from the same environment set up.

I feel I could have done better to organise communication in the team to ensure everyone has a working set up for testing, (which has been discussed for next weeks meeting) as this was something that could have been resolved within the first couple days and something that needs to be resolved early since ensuring a large team of people are set up correctly takes time.

Conclusion

In conclusion it's clear Learning a new framework while working on it is often a struggle as it is a big learning curve however it's a difficulty I have found enjoyable and rewarding as upskilling myself to a new framework and comparing code is while becoming a better software engineer is what I am here for.

Moving forward I will need to assign more time into researching the framework so I can improve further on my skillset for this project and assist others who are falling behind and have another attempt at improving communications with team members to work closer together.