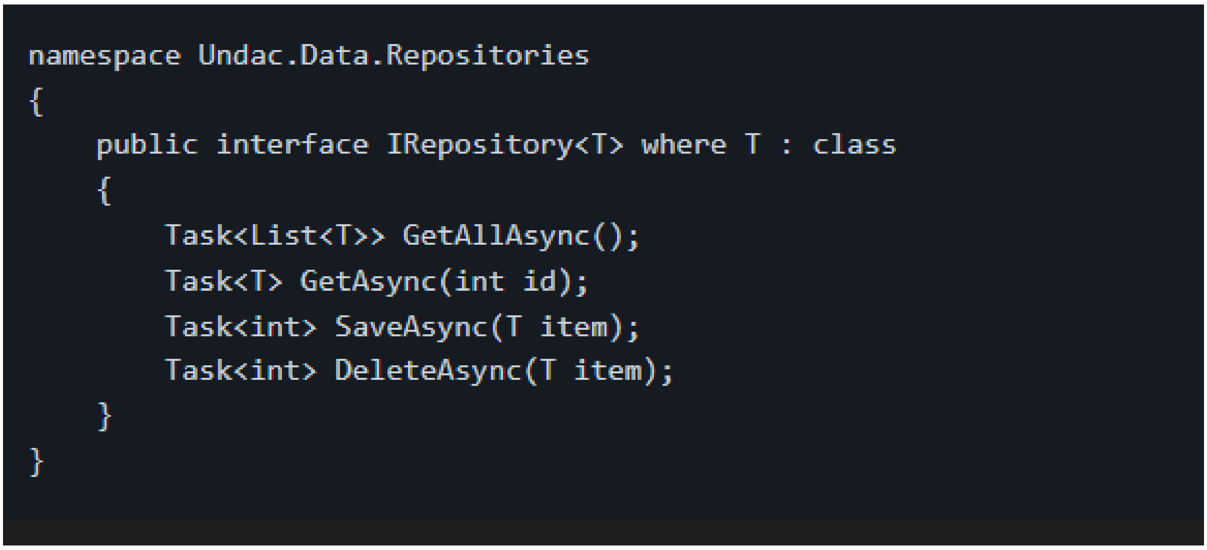
Project work 1

Summary of issue

This week I returned to the previous issue Equipment Types to update the codes functionality, update CRUD functions and to
introduce a repository pattern as each issue has a different model, there is a lot of code repetition. This repetion violates the
DRY(Don't Repeat Yourself) Principle as with each model comes a set of CRUD functions from other issues. The repository pattern
will stop this from happeneing.

The repository was pulled from PR#51 as a collegue already created the repository pattern it just needed implemented and CRUD
updated to fit this pattern

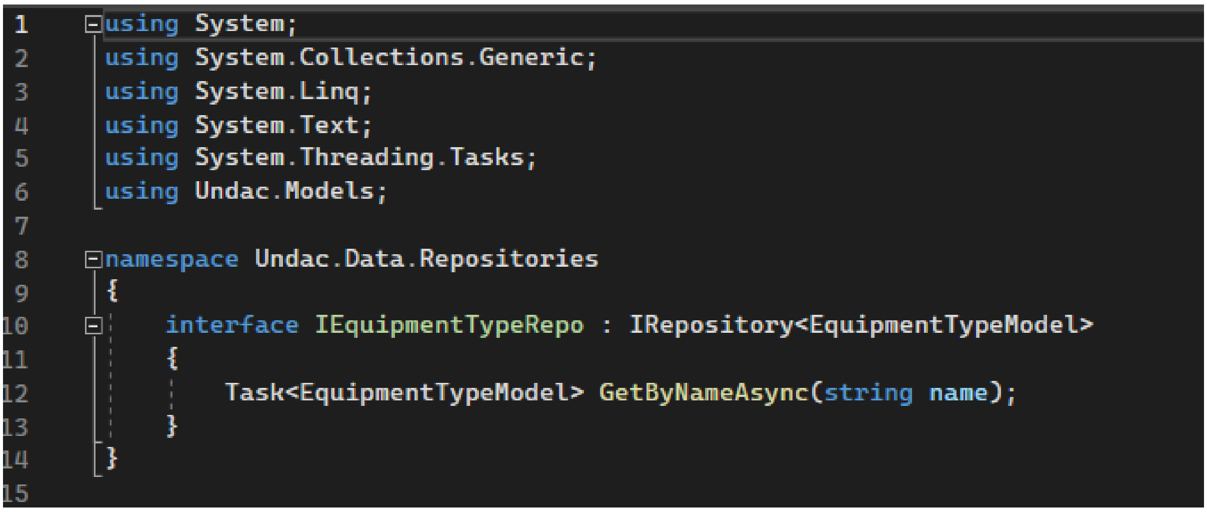
In figure 1. you'll see the interface that has all required CRUD methods. The T stands as a general token reciever of a class type
so that multiple model objects of different type can be accepted



```
namespace Undac.Data.Repositories
{
    public interface IRepository<T> where T : class
    {
        Task<List<T>> GetAllAsync();
        Task<T> GetAsync(int id);
        Task<int> SaveAsync(T item);
        Task<int> DeleteAsync(T item);
    }
}
```

***Figure 1**

Since the project uses an SQLite local database this is how the pattern looks implemented
First, an interface was created and as can be seen from figure 2 the Equipment Types implements the repositories interface where the CRUD
functions a located and passes the model class as the token T and gets the name returned.

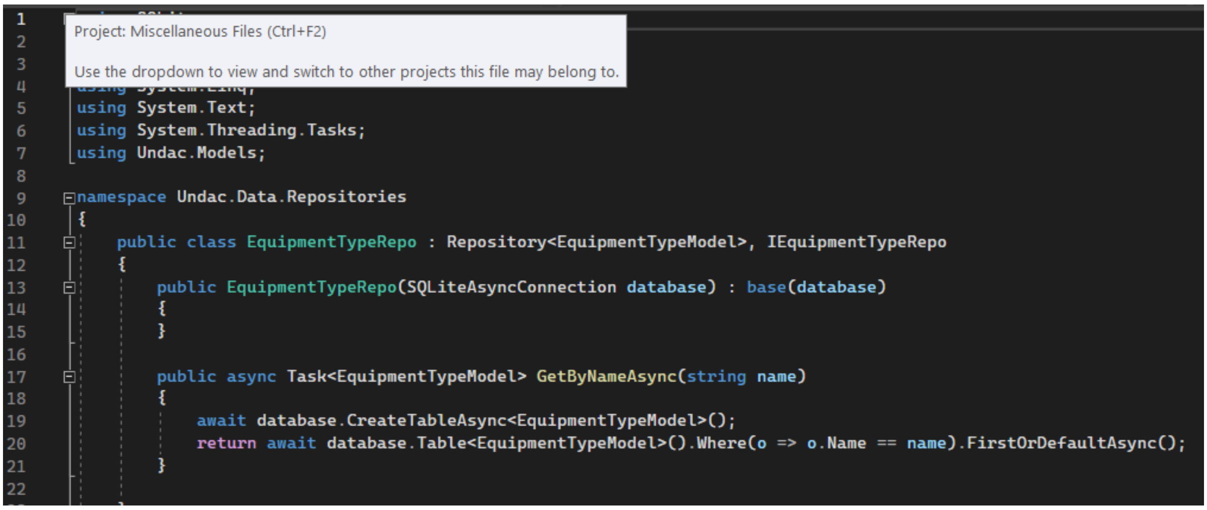


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Undac.Models;

namespace Undac.Data.Repositories
{
    interface IEquipmentTypeRepo : IRepository<EquipmentTypeModel>
    {
        Task<EquipmentTypeModel> GetByNameAsync(string name);
    }
}
```

***Figure 2**

After creating the interface its time to implement the repository to connect to the database and retrive the name of the equipment types.



Project: Miscellaneous Files (Ctrl+F2)

Use the dropdown to view and switch to other projects this file may belong to.

```
using System.Text;
using System.Threading.Tasks;
using Undac.Models;

namespace Undac.Data.Repositories
{
    public class EquipmentTypeRepo : Repository<EquipmentTypeModel>, IEquipmentTypeRepo
    {
        public EquipmentTypeRepo(SQLiteAsyncConnection database) : base(database)
        {
        }

        public async Task<EquipmentTypeModel> GetByNameAsync(string name)
        {
            await database.CreateTableAsync<EquipmentTypeModel>();
            return await database.Table<EquipmentTypeModel>().Where(o => o.Name == name).FirstOrDefaultAsync();
        }
```

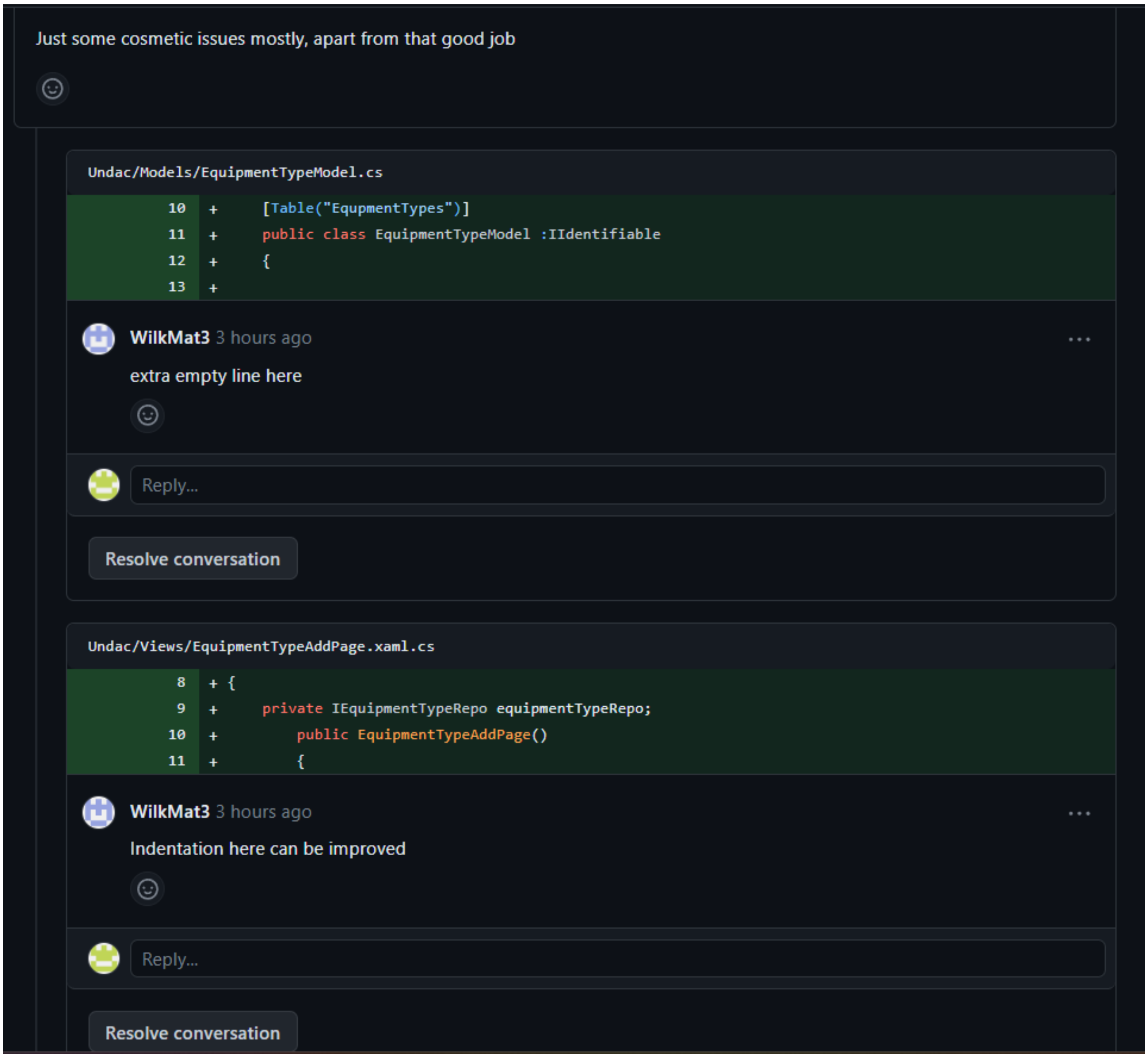
***Figure 3**

Test

Unfortuneately i was unable to produce the appropriate tests to test the methods within the project.
The testing that was tried to be implemented was NUnit tests as the team agreed to use this testing framework however I do have knowledge of
this testing framework i had difficulty producing the test methods.

Code Review

The Reviews on my code showed there were conflicts within the appshell.xaml file this was due to conflicting navigation design.
The reviews also state that I have broken standard c# conventions when it comes the code layout and indenting. In figures 4 & 5 you can see
the reviews left.



Just some cosmetic issues mostly, apart from that good job

Undac/Models/EquipmentTypeModel.cs

```
+     [Table("EqupmentTypes")]
+     public class EquipmentTypeModel :IIdentifiable
+     {
+
```

WilkMat3 3 hours ago

extra empty line here

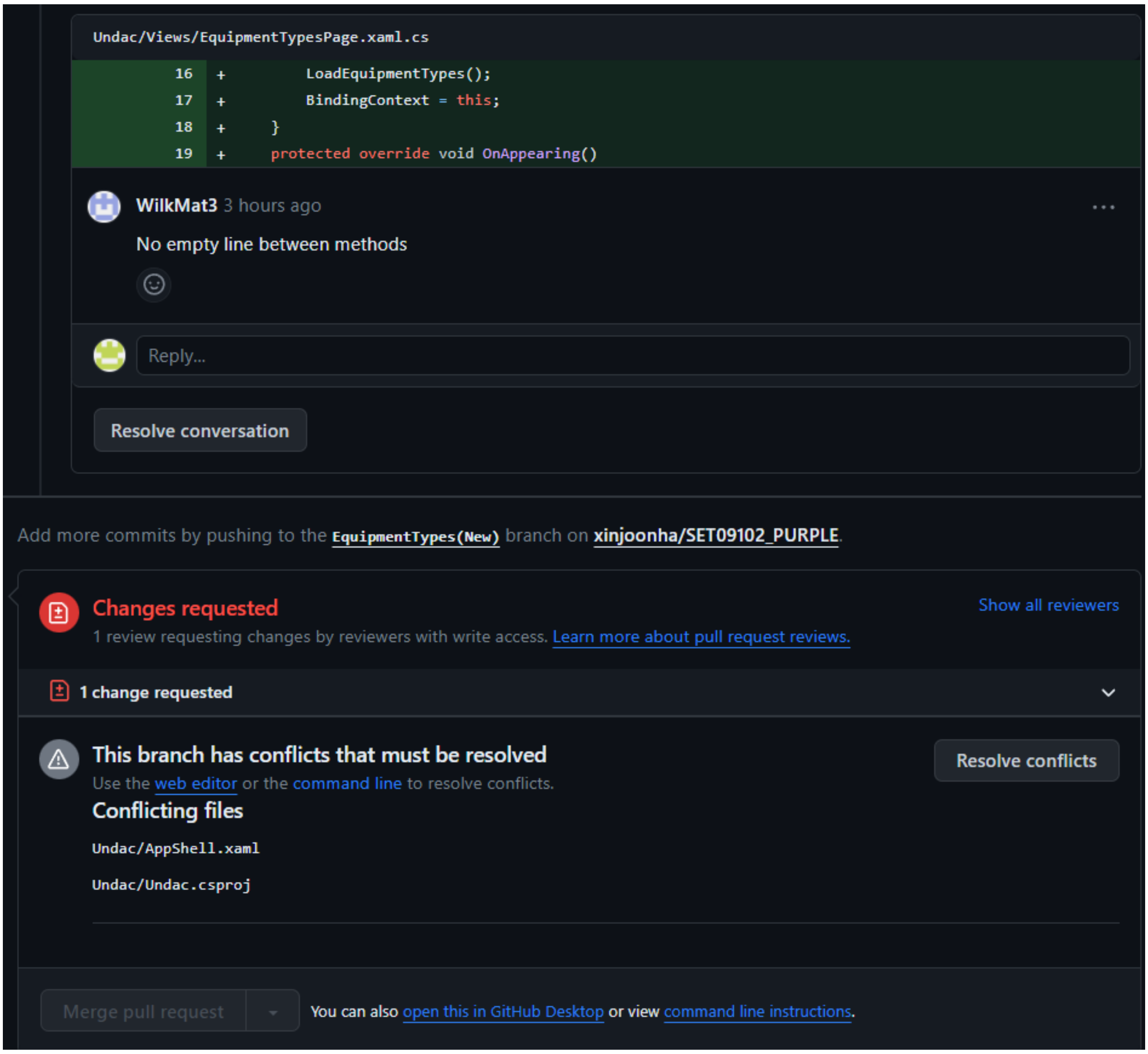
Undac/Views/EquipmentTypeAddPage.xaml.cs

```
+ {
+     private IEquipmentTypeRepo equipmentTypeRepo;
+         public EquipmentTypeAddPage()
+         {
```

WilkMat3 3 hours ago

Indentation here can be improved

***Figure 4**



Undac/Views/EquipmentTypesPage.xaml.cs

```
+         LoadEquipmentTypes();
+         BindingContext = this;
+     }
+     protected override void OnAppearing()
```

WilkMat3 3 hours ago

No empty line between methods

Add more commits by pushing to the EquipmentTypes(New) branch on xinjoonha/SET09102_PURPLE.

Changes requested

1 review requesting changes by reviewers with write access. Learn more about pull request reviews.

1 change requested

This branch has conflicts that must be resolved

Use the web editor or the command line to resolve conflicts.

Conflicting files

Undac/AppShell.xaml

Undac/Undac.csproj

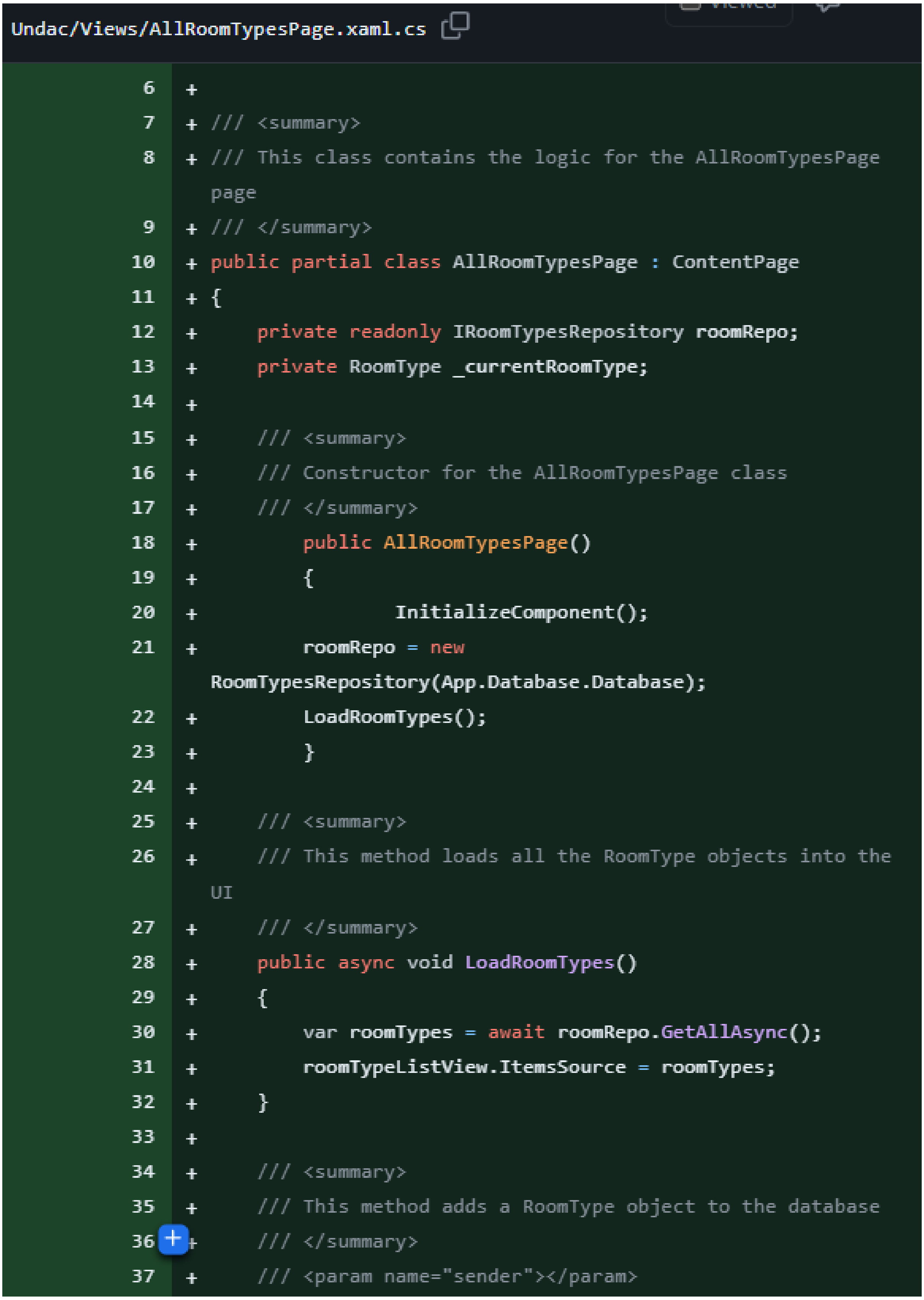
You can also open this in GitHub Desktop or view command line instructions.

***Figure 5**

The conflict within the Undac.csproj is still currently under investigation as there is no obvious reason for the conflict.

Reviewing code

Upon Reviewing PR#55 which is to recruit pool experts it was found that there were no conflicts or issues with the code it used the repository pattern
so the code followed the DRY principle. The code also followed standard c# conventions to a high standard with clear naming conventions, correct
formatting and being easily readable.
The commenting on the code for documentation was quite good and extensive but brief and informative as can be seen in figure 6.



Undac/Views/AllRoomTypesPage.xaml.cs

```
+
+ /// <summary>
+ /// This class contains the logic for the AllRoomTypesPage page
+ /// </summary>
+ public partial class AllRoomTypesPage : ContentPage
+ {
+     private readonly IRoomTypesRepository roomRepo;
+     private RoomType _currentRoomType;
+
+     /// <summary>
+     /// Constructor for the AllRoomTypesPage class
+     /// </summary>
+         public AllRoomTypesPage()
+         {
+                 InitializeComponent();
+         roomRepo = new RoomTypesRepository(App.Database.Database);
+         LoadRoomTypes();
+         }
+
+     /// <summary>
+     /// This method loads all the RoomType objects into the UI
+     /// </summary>
+     public async void LoadRoomTypes()
+     {
+         var roomTypes = await roomRepo.GetAllAsync();
+         roomTypeListView.ItemsSource = roomTypes;
+     }
+
+     /// <summary>
+     /// This method adds a RoomType object to the database
+     /// </summary>
+     /// <param name="sender"></param>
```

***Figure 6**

Reflection

This week has been tough as the gap in knowledge between me and the rest of the group due to inexperience has left me having to ask for their help on

several occasions to go over the program and what tools i can use to improve my learning, however, i have learned quite a lot this week and been able
close the gap a little. The reasoning for not being able to produce the test methods are that between catching up on the programming language and using
frameworks that i have never used before and trying to complete the tasks everyweek there was simply not enough time to complete the test environment.

As for teamwork the team have engaged more this week than ever the team have about 5 or 6 people that have extra knowledge and are using it to better
the workflow but the team are far from perfect however they seem to have been more organsied than ever.

I believe the group as a whole myself included are not advancing enough within the project but I think this has to do with lack of knowledge in the same
areas as myself and the stress of having to try and keep up has them disheartened and therefore less engaged.