

## API Documentation

### Version 1

#### Database:

Setup models to represent the different tables in the database.

#### Users

```
2022 API_Studio_2022.Models.Users
namespace API_Studio_2022.Models
{
    2 references
    public class Users
    {
        0 references
        public int UserID { get; set; }

        0 references
        public string userName { get; set; }

        0 references
        public string password { get; set; }

        0 references
        public string securityQuestion { get; set; }

        0 references
        public string contactInfo { get; set; }

        0 references
        public ICollection<Roles> RoleID { get; set; }
    }
}
```

#### User Analytics

```
2022 API_Studio_2022.Models.UserAnalytics
namespace API_Studio_2022.Models
{
    0 references
    public class UserAnalytics
    {
        0 references
        public DateTime fileUpdated { get; set; }

        0 references
        public int filesRated { get; set; }

        0 references
        public int filesReported { get; set; }

        0 references
        public int passswordReset { get; set; }

        0 references
        public int logins { get; set; }

        0 references
        public ICollection<Users> UserID { get; set; }
    }
}
```

## Roles

```
1 namespace API_Studio_2022.Models
2 {
3     1 reference
4     public class Roles
5     {
6         0 references
7         public INamedRouter RoleID { get; set; }
8
9         0 references
10        public string description { get; set; }
11    }
12 }
```

## Files

```
3 public class Files
4 {
5
6     0 references
7     public int FileID { get; set; }
8
9     0 references
10    public DateTime dateChanged { get; set; }
11
12    0 references
13    public string subject { get; set; }
14
15    0 references
16    public int grade { get; set; }
17
18    0 references
19    public string keyWord { get; set; }
20
21    0 references
22    public int rating { get; set; }
23
24    0 references
25    public string storageLoc { get; set; }
26
27    0 references
28    public ICollection<Users> UserID { get; set; }
29 }
```

## File Changes

```
1 namespace API_Studio_2022.Models
2 {
3     0 references
4     public class FAQ
5     {
6         0 references
7         public int QuestID { get; set; }
8
9         0 references
10        public string questionDesc { get; set; }
11
12        0 references
13        public string awnser { get; set; }
14    }
15 }
```

## FAQ

```
1 namespace API_Studio_2022.Models
2 {
3     0 references
4     public class FileChanges
5     {
6         0 references
7         public int ChangeID { get; set; }
8
9         0 references
10        public DateTime dateChanged { get; set; }
11
12        0 references
13        public string changeDesc { get; set; }
14
15        0 references
16        public ICollection<Files> FileID { get; set; }
17    }
18 }
```

We also link the data tables with their many to one relationship to insure all the tables can interact and store data correctly.

## Users

```
0 references  
public ICollection<Roles> RoleID { get; set; }
```

## User Analytics

```
0 references  
public ICollection<Users> UserID { get; set; }
```

## Roles

None

## Files

```
0 references  
public ICollection<Users> UserID { get; set; }
```

## File Changes

```
0 references  
public ICollection<Files> FileID { get; set; }
```

## FAQ

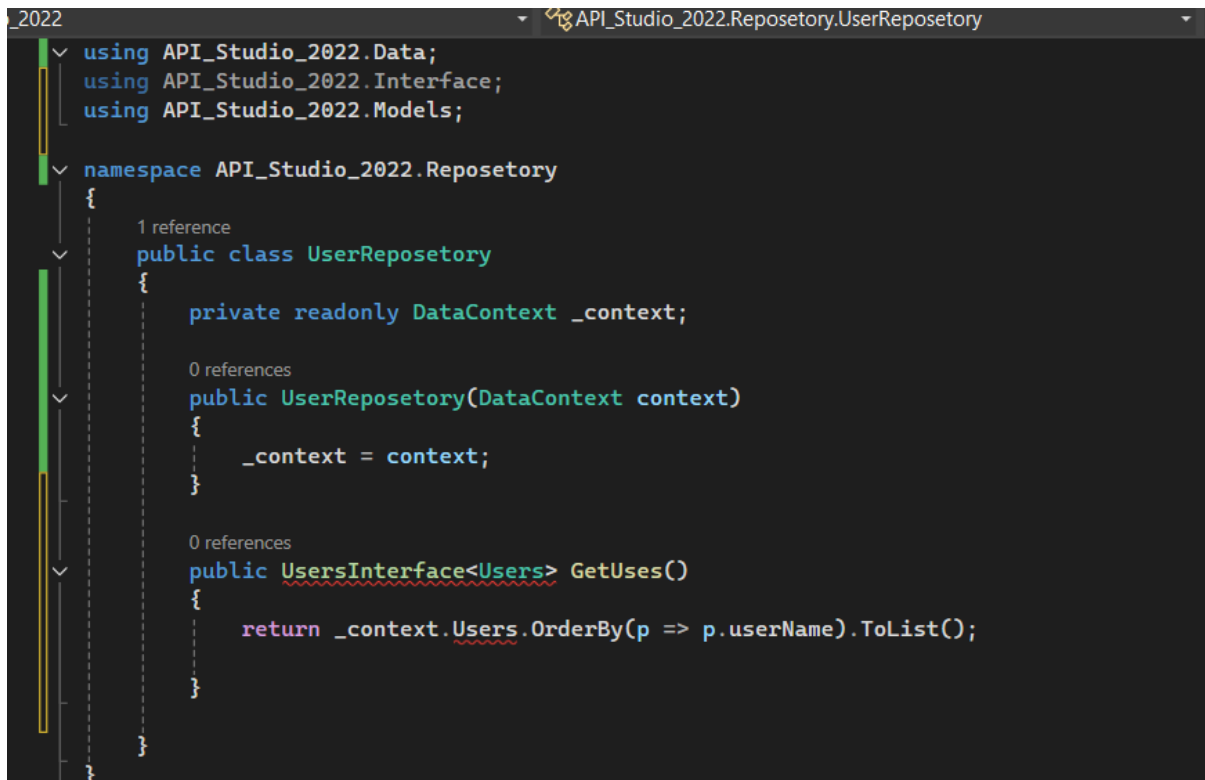
None

As an example, I will post a code comment for a chase of Many to Many relationships, but we have no such relationships in our data

```
0 references  
public class DataContext : DbContext  
{  
    // Example if a many to many relationship exists  
    /*  
    public DataContext(DbContextOptions<DataContext> options) : base(options)  
    {  
    }  
    public DbSet<Users> Users { get; set; }  
    public DbSet<FAQ> NewQuestions { get; set; }  
    public DbSet<FileChanges> FilesChanged { get; set; }  
    */  
}
```

## The Calls of the API:

### Example of a Method

A screenshot of a code editor showing the implementation of the `UserRepository` class. The code is in C# and is part of the `API_Studio_2022.Repository` namespace. It includes using statements for `API_Studio_2022.Data`, `API_Studio_2022.Interface`, and `API_Studio_2022.Models`. The class `UserRepository` has a private readonly `DataContext` property `_context`. It has a constructor `UserRepository(DataContext context)` that initializes `_context`. It also has a public method `GetUses()` that returns a list of users ordered by their username. The code is color-coded and has a sidebar on the left showing the file explorer.

```
using API_Studio_2022.Data;
using API_Studio_2022.Interface;
using API_Studio_2022.Models;

namespace API_Studio_2022.Repository
{
    1 reference
    public class UserRepository
    {
        private readonly DataContext _context;

        0 references
        public UserRepository(DataContext context)
        {
            _context = context;
        }

        0 references
        public UsersInterface<Users> GetUses()
        {
            return _context.Users.OrderBy(p => p.userName).ToList();
        }
    }
}
```

### Get & Read Methods

Different methods used include the Get and read methods where we use the command Get. As shown in the example above this allows us to fetch data from a certain source in this case it is the Users table. We can thus change the `return` if we want to change the type of output we want to achieve.

We also have the functionality for the following method types to be used if needed:

1. Post & Create Methods
2. Update & Put Methods
3. Delete Methods