Solution Document

في البداية نقوم بإنشاء المشروع الأساسي ومشروع الاختبار ضمن sln:

```
PS D:\Study\Software_Engeneering_2\LoanApp> dotnet new sln -n LoanApp
The template "Solution File" was created successfully.

PS D:\Study\Software_Engeneering_2\LoanApp> dotnet new classlib -n LoanApp.Core
The template "Class Library" was created successfully.

Processing post-creation actions...
Restoring D:\Study\Software_Engeneering_2\LoanApp\LoanApp.Core\LoanApp.Core.csproj:
  Determining projects to restore...
  Restored D:\Study\Software_Engeneering_2\LoanApp\LoanApp.Core\LoanApp.Core.csproj (in
   133 ms).
Restore succeeded.


PS D:\Study\Software_Engeneering_2\LoanApp> dotnet new xunit -n LoanApp.Tests
The template "xUnit Test Project" was created successfully.

  PS D:\Study\Software_Engeneering_2\LoanApp> dotnet sln add LoanApp.Core/LoanApp.Core.cspr
  oj
  Project `LoanApp.Core\LoanApp.Core.csproj` added to the solution.
  PS D:\Study\Software_Engeneering_2\LoanApp> dotnet sln add LoanApp.Tests/LoanApp.Tests.cs
  proj
  Project `LoanApp.Tests\LoanApp.Tests.csproj` added to the solution.
  PS D:\Study\Software_Engeneering_2\LoanApp> dotnet  add LoanApp.Tests/LoanApp.Tests.cspro
  j reference LoanApp.Core/LoanApp.Core.csproj
  Reference `..\LoanApp.Core\LoanApp.Core.csproj` added to the project.
```

بعد ذلك نقوم بكتابة الكود التالي في LoanEvaluator.cs:

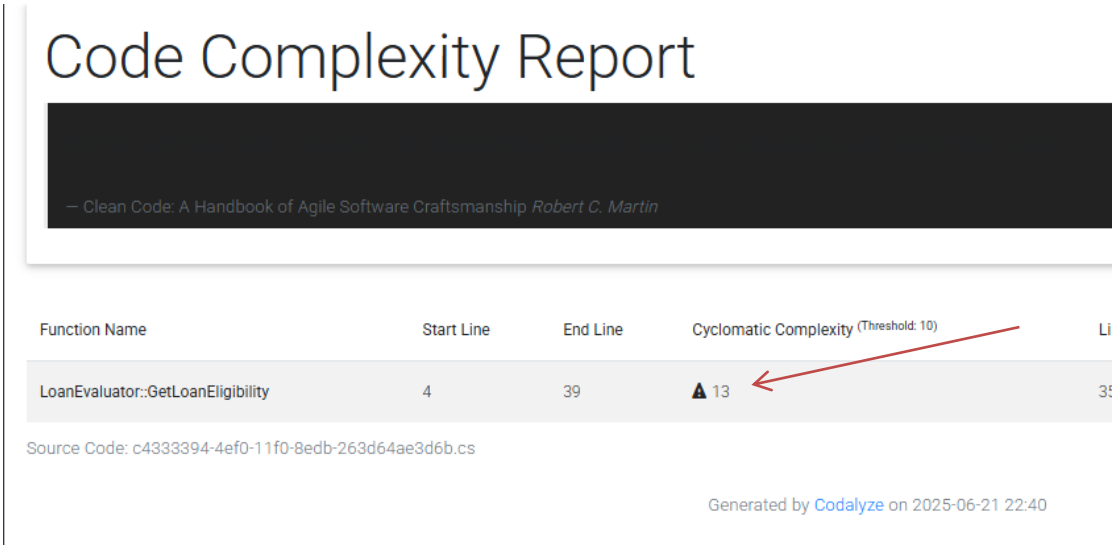
LoanApp.Core > LoanEvaluator.cs > LoanEvaluator > GetLoanEligibility

```
namespace LoanApp;
1 reference
public class LoanEvaluator
{
    1 reference
    public static string GetLoanEligibility(int income, bool hasJob, int creditScore, int dependents, bool ownsHouse)
    {
        if (income < 2000)
            return "Not Eligible";

        if (hasJob)
        {
            if (creditScore >= 700)
            {
                if (dependents == 0)
                    return "Eligible";
                else if (dependents <= 2)
                    return "Review Manually";
                else
                    return "Not Eligible";
            }
            else if (creditScore >= 600)
            {
                if (ownsHouse)
                    return "Review Manually";
                else
                    return "Not Eligible";
            }
            else
                return "Not Eligible";
        }
        else
        {
            if (creditScore >= 750 && income > 5000 && ownsHouse)
                return "Eligible";
            else if (creditScore >= 650 && dependents == 0)
                return "Review Manually";
            else
                return "Not Eligible";
        }
    }
}
```

لنقوم بحساب التعقيد Cyclomatic Complexity :

نلاحظ التعقيد المرتفع الذي يبلغ 13 وللتأكد من ذلك نستخدم الأداة codalyze:



لحل المشكلة وتخفيض التعقيد نطبق ال refactoring وذلك لفصل الوظائف:

```
namespace LoanApp.Core;

1 reference
public static class LoanEvaluator
{
    1 reference
    public static string GetLoanEligibility(int income, bool hasJob, int creditScore, int dependents, bool ownsHouse)
    {
        if (IsLowIncome(income))
            return "Not Eligible";

        if (hasJob)
            return EvaluateEmployed(creditScore, dependents, ownsHouse);
        else
            return EvaluateUnEmployed(income, creditScore, dependents, ownsHouse);
    }

    1 reference
    private static bool IsLowIncome(int income) => income < 2000;

    1 reference
    private static string EvaluateEmployed(int creditScore, int dependents, bool ownsHouse)
    {
        if (creditScore >= 700)
        {
            if (dependents == 0) return "Eligible";
            else if (dependents <= 2) return "Review Manually";
            return "Not Eligible";
        }

        else if (creditScore >= 600)
            return ownsHouse ? "Review Manually" : "Not Eligible";

        return "Not Eligible";
    }

    1 reference
    private static string EvaluateUnEmployed(int income, int creditScore, int dependents, bool ownsHouse)
    {
        if (creditScore >= 750 && income > 5000 && ownsHouse)
            return "Eligible";

        else if (creditScore >= 650 && dependents == 0)
            return "Review Manually";

        return "Not Eligible";
    }
}
```

1

2

CC = D + 1

CC = 2 + 1 = 3

Code Complexity Report

— Clean Code: A Handbook of Agile Software Craftsmanship *Robert C. Martin*

Function Name	Start Line	End Line	Cyclomatic Complexity (Threshold: 10)
LoanEvaluator::GetLoanEligibility	5	14	3

Source Code: f68c3214-4ef5-11f0-8edb-263d64ae3d6b.cs

Generated by Codalyze on 2025-06-2

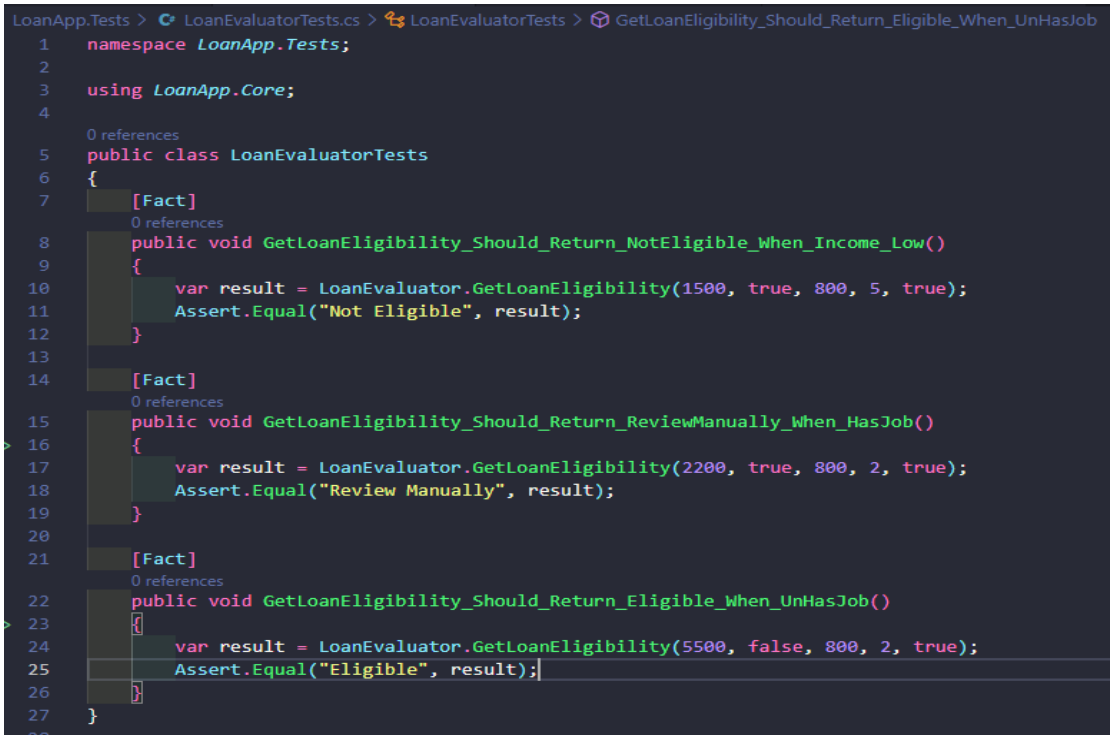
العمل تم على مبدأ مثال المحاضرة وهو Single Responsibility لقد قمنا بتخفيض التعقيد من خلال فصل التابع الرئيسي إلى ثلاث حالات وهي income ,has job, not has job

ومن ثم داخل كل منها قمنا باستدعاء التوابع المساعدة التي وضعناها فيها ال logic وهي:

- IsLowIncome
- EvaluateEmploye
- EvaluateUnEmploye

وهكذا وصلنا بتعقيد التابع الرئيسي إلى القيمة 3.

نكتب هنا توابع اختبار التابع الرئيسي الذي أصبح بالشكل التالي بعد نقل التوابع المساعدة إلى ملف ثان:



قمنا بكتابة ثلاث توابع اختبار أولها لحالة LowIncome والثانية HasJob والثالثة UnHasJob ونلاحظ تحقق العلاقة التالية:

- **عدد حالات الاختبار الدنيا المطلوبة ≤ قيمة CC**

بعد ذلك ننتقل إلى العمل مع التوابع المساعدة التالية...

نقوم بإنشاء ملف LoanEvaluatorHelpersTests.cs في مشروع الاختبار ونكتب فيه بعض توابع الاختبار منها:

```
LoanApp.Core > LoanEvaluatorHelpers.cs > LoanEvaluatorHelpers
namespace LoanApp.Core;

6 references
public static class LoanEvaluatorHelpers
{
    3 references
    public static string EvaluateEmployed(int creditScore, int dependents, bool ownsHouse)
    {
        if (creditScore >= 700)
        {
            if (dependents == 0) return "Eligible";
            else if (dependents <= 2) return "Review Manually";
            return "Not Eligible";
        }

        else if (creditScore >= 600)
            return ownsHouse ? "Review Manually" : "Not Eligible";

        return "Not Eligible";
    }

    3 references
    public static string EvaluateUnEmployed(int income, int creditScore, int dependents, bool ownsHouse)
    {
        if (creditScore >= 750 && income > 5000 && ownsHouse)
            return "Eligible";

        else if (creditScore >= 650 && dependents == 0)
            return "Review Manually";

        return "Not Eligible";
    }
}
```

```
public static string EvaluateEmployed(int creditScore, int dependents, bool ownsHouse)
{
    if (creditScore >= 700)          1
    {
        if (dependents == 0) return "Eligible";                2
        else if (dependents <= 2) return "Review Manually";    3
        return "Not Eligible";
    }

    else if (creditScore >= 600)     4
        return ownsHouse ? "Review Manually" : "Not Eligible";  5

    return "Not Eligible";
}
```

CC = D + 1

CC = 5 + 1 = 6

Code Complexity Report

— Clean Code: A Handbook of Agile Software Craftsmanship *Robert C. Martin*

Function Name	Start Line	End Line	Cyclomatic Complexity (Threshold: 10)
LoanEvaluatorHelpers::EvaluateEmployed	5	18	6
LoanEvaluatorHelpers::EvaluateUnEmployed	20	29	6

Source Code: 90859744-5067-11f0-8edb-263d64ae3d6b.cs

Generated by Codalyze on 2025-06-23 19:23

نلاحظ أن في التابع المساعد الأول EvaluateEmploye قيمة CC = 6 وبالتالي نحتاج على الأقل 6 توابع اختبار وهي:

```
[Fact]
public void EvaluateEmployed_Should_ReturnEligible_When_ScoreOver700AndNoDependents()
{
    var result = LoanEvaluatorHelpers.EvaluateEmployed(800, 0, true);
    Assert.Equal("Eligible", result);
}

[Fact]
public void EvaluateEmployed_Should_ReturnReviewManually_When_ScoreOver700AndLessOrEqualTwoDependents()
{
    var result = LoanEvaluatorHelpers.EvaluateEmployed(800, 2, true);
    Assert.Equal("Review Manually", result);
}

[Fact]
public void EvaluateEmployed_Should_ReturnNotEligible_When_ScoreOver700AndMoreTwoDependents()
{
    var result = LoanEvaluatorHelpers.EvaluateEmployed(800, 5, true);
    Assert.Equal("Not Eligible", result);
}
```

```
28      [Fact]
29      public void EvaluateEmployed_Should_ReturnReviewManually_When_ScoreOver600AndOwnsHouse()
30      {
31          var result = LoanEvaluatorHelpers.EvaluateEmployed(650, 5, true);
32          Assert.Equal("Review Manually", result);
33      }
34
35      [Fact]
36      public void EvaluateEmployed_Should_ReturnNotEligible_When_ScoreOver600AndNotOwnsHouse()
37      {
38          var result = LoanEvaluatorHelpers.EvaluateEmployed(600, 5, false);
39          Assert.Equal("Not Eligible", result);
40      }
41
42      [Fact]
43      public void EvaluateEmployed_Should_ReturnNotEligible_When_PreviousIsFalse()
44      {
45          var result = LoanEvaluatorHelpers.EvaluateEmployed(500, 5, false);
46          Assert.Equal("Not Eligible", result);
47      }
```

أما في التابع المساعد الثاني EvaluateUnEmploye:

```
public static string EvaluateUnEmployed(int income, int creditScore, int dependents, bool ownsH
{
    if (creditScore >= 750 && income > 5000 && ownsHouse)
        return "Eligible";

    else if (creditScore >= 650 && dependents == 0)
        return "Review Manually";

    return "Not Eligible";
}
```

1 2 3

4 5

CC = D + 1
CC = 5 + 1 = 6

Code Complexity Report

— Clean Code: A Handbook of Agile Software Craftsmanship *Robert C. Martin*

Function Name	Start Line	End Line	Cyclomatic Complexity (Threshold: 10)
LoanEvaluatorHelpers::EvaluateEmployed	5	18	6
LoanEvaluatorHelpers::EvaluateUnEmployed	20	29	6

Source Code: 90859744-5067-11f0-8edb-263d64ae3d6b.cs

Generated by Codalyze on 2025-06-23 19:23

قيمة CC = 6 لذلك أيضا يوجد 6 حالات اختبار على الأقل وهي:

```
51      // EvaluateUnEmployed Testing
52      [Fact]
53      public void EvaluateUnEmployed_Should_ReturnEligible_When_ScoreOver750AndInComeOver5000AndOwnsHouse()
54      {
55          var result = LoanEvaluatorHelpers.EvaluateUnEmployed(5500, 1000, 2, true);
56          Assert.Equal("Eligible", result);
57      }
58
59      [Fact]
60      public void EvaluateUnEmployed_Should_ReturnReviewManually_When_ScoreOver650AndNoDependents()
61      {
62          var result = LoanEvaluatorHelpers.EvaluateUnEmployed(3000, 1000, 0, true);
63          Assert.Equal("Review Manually", result);
64      }
65
66      [Fact]
67      public void EvaluateUnEmployed_Should_ReturnNotEligible_When_ScoreLess650()
68      {
69          var result = LoanEvaluatorHelpers.EvaluateUnEmployed(5000, 500, 0, true);
70          Assert.Equal("Not Eligible", result);
71      }
72
```

```
73      [Fact]
74      public void EvaluateUnEmployed_Should_ReturnNotEligible_When_ScoreOver750AndNotOwnsHouseAndOneDependent()
75      {
76          var result = LoanEvaluatorHelpers.EvaluateUnEmployed(5000, 1000, 1, false);
77          Assert.Equal("Not Eligible", result);
78      }
79
80      [Fact]
81      public void EvaluateUnEmployed_Should_ReturnNotEligible_When_ScoreOver750AndIncomeLessThan5000AndOneDependent()
82      {
83          var result = LoanEvaluatorHelpers.EvaluateUnEmployed(4000, 1000, 1, true);
84          Assert.Equal("Not Eligible", result);
85      }
86
87      [Fact]
88      public void EvaluateUnEmployed_Should_ReturnNotEligible_When_ScoreOver650AndOneDependent()
89      {
90          var result = LoanEvaluatorHelpers.EvaluateUnEmployed(5000, 670, 1, true);
91          Assert.Equal("Not Eligible", result);
92      }
93
```

الان لو قمنا بتنفيذ dotnet test سنحصل على النتيجة وهي نجاح جميع حالات الاختبار:

```
  Determining projects to restore...
  All projects are up-to-date for restore.
  LoanApp.Core -> D:\Study\Software_Engeneering_2\LoanApp\LoanApp.Core\bin\Debug\net8.0\LoanApp.Core.dll
  LoanApp.Tests -> D:\Study\Software_Engeneering_2\LoanApp\LoanApp.Tests\bin\Debug\net8.0\LoanApp.Tests.dll
Test run for D:\Study\Software_Engeneering_2\LoanApp\LoanApp.Tests\bin\Debug\net8.0\LoanApp.Tests.dll (.NETCoreApp,V
VSTest version 17.11.1 (x64)

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed!  - Failed:     0, Passed:    15, Skipped:     0, Total:    15, Duration: 75 ms - LoanApp.Tests.dll (net8.0)
```