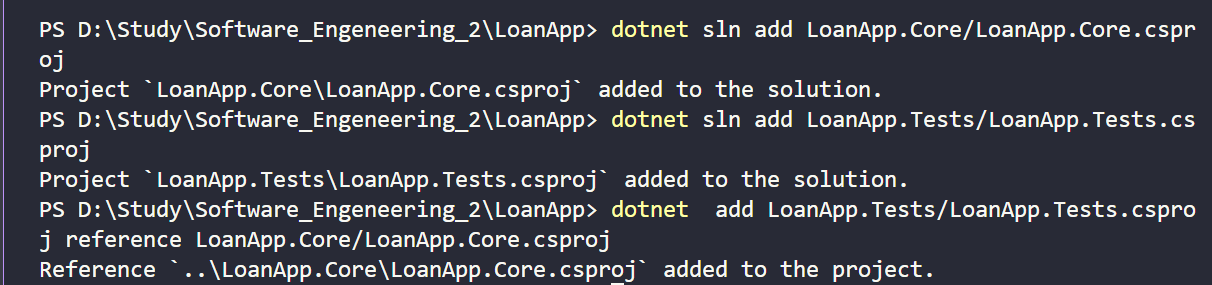
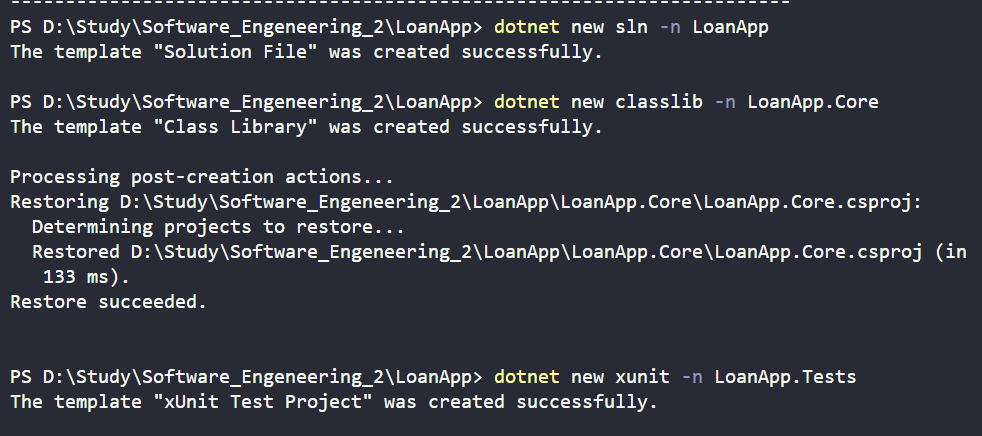
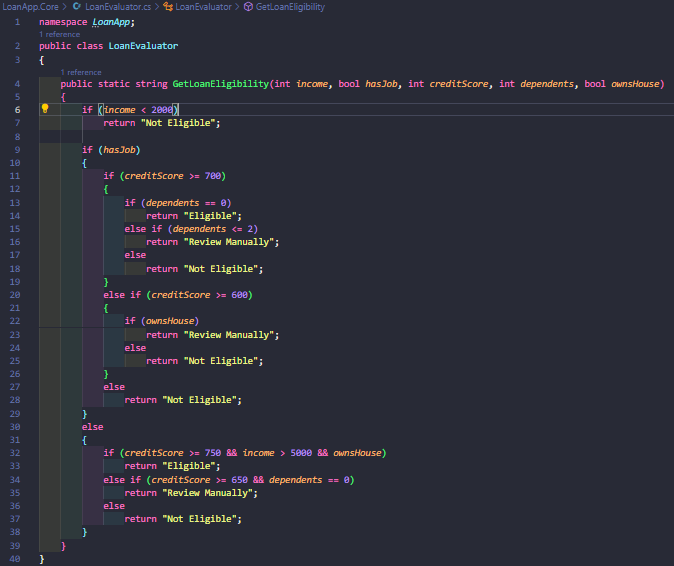
**Solution Document**

في البداية نقوم بإنشاء المشروع الأساسي ومشروع الاختبار ضمن sln:



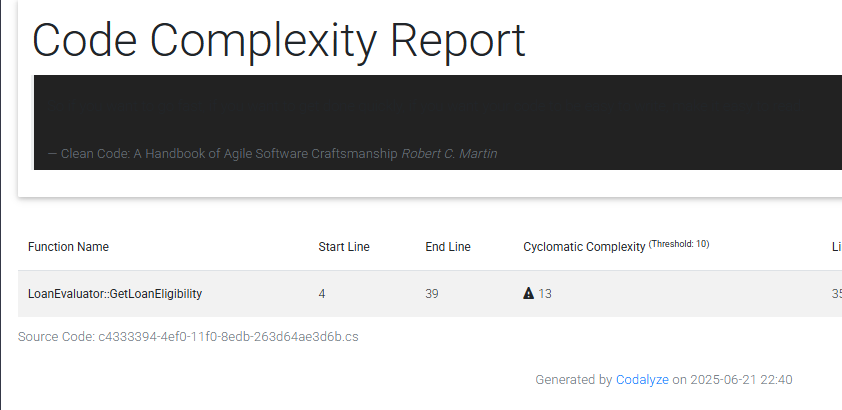
بعد ذلك نقوم بكتابة الكود التالي في :LoanEvaluator.cs



لنقوم بحساب التعقيد Cyclomatic Complexity :

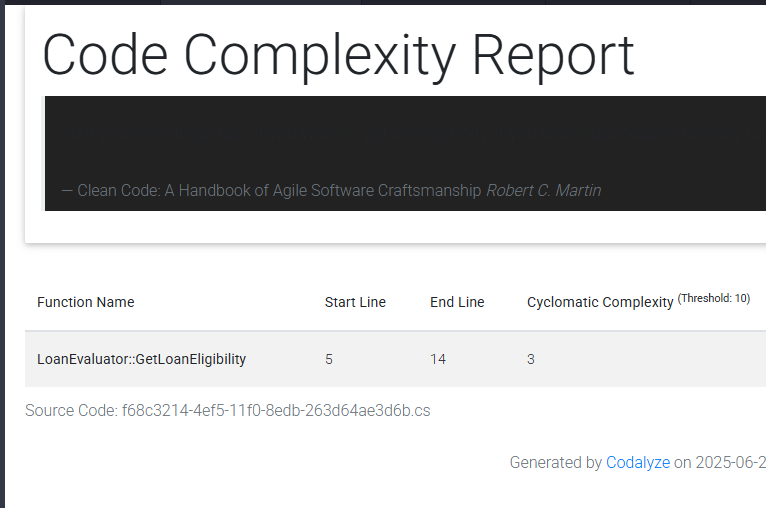


نلاحظ التعقيد المرتفع الذي يبلغ 13 وللتأكد من ذلك نستخدم الأداة codalyze:



لحل المشكلة وتخفيض التعقيد نطبق ال refactoring وذلك لفصل الوظائف:





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:



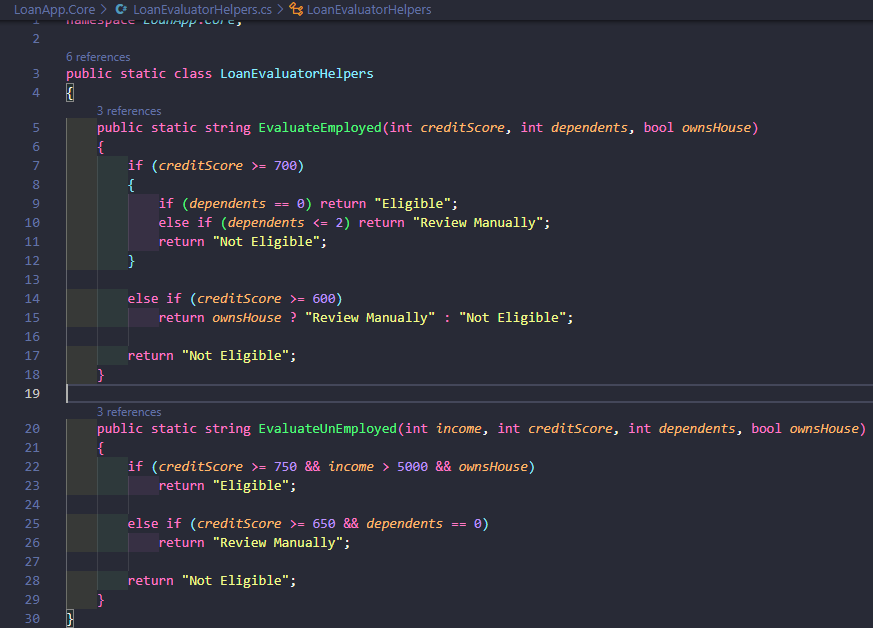


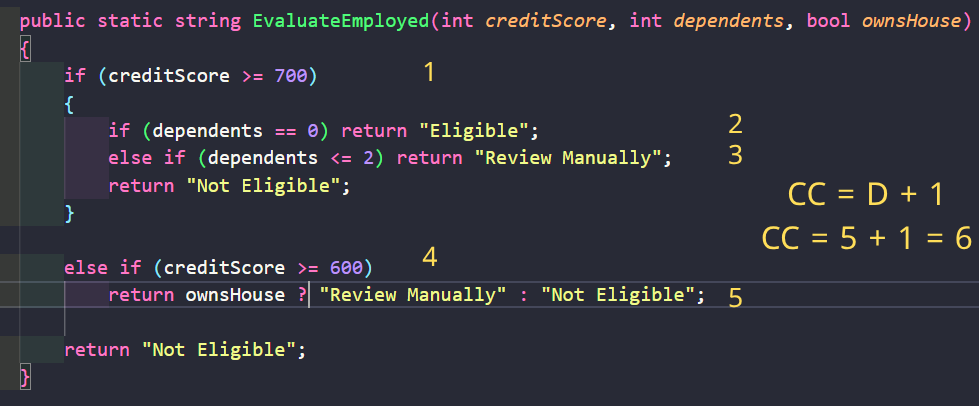
قمنا بكتابة ثلاث توابع اختبار أولها لحالة LowIncome والثانية HasJob والثالثة UnHasJob ونلاحظ تحقق العلاقة التالية:

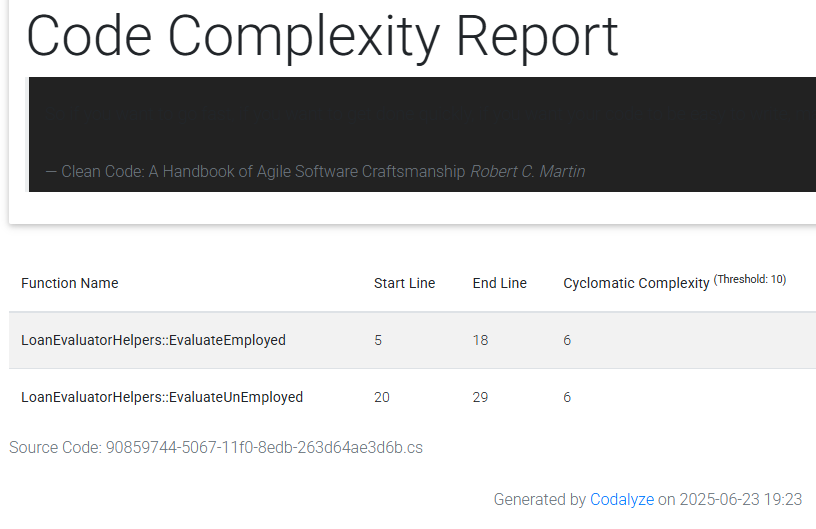


بعد ذلك ننتقل إلى العمل مع التوابع المساعدة التالية...

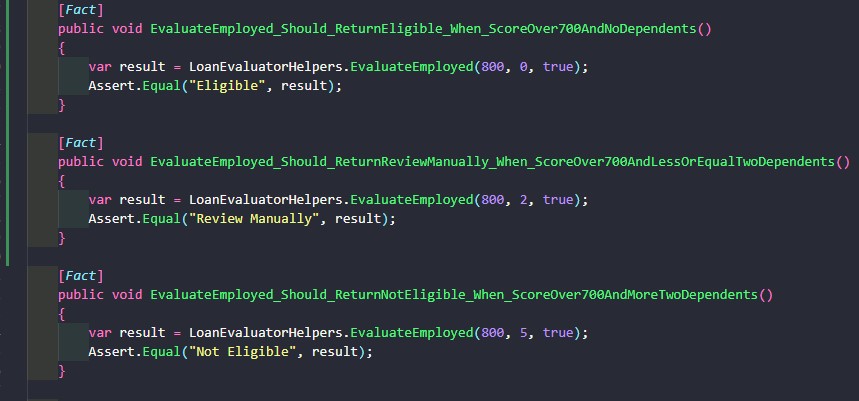
نقوم بإنشاء ملف LoanEvaluatorHelpersTests.cs في مشروع الاختبار ونكتب فيه بعض توابع الاختبار منها:

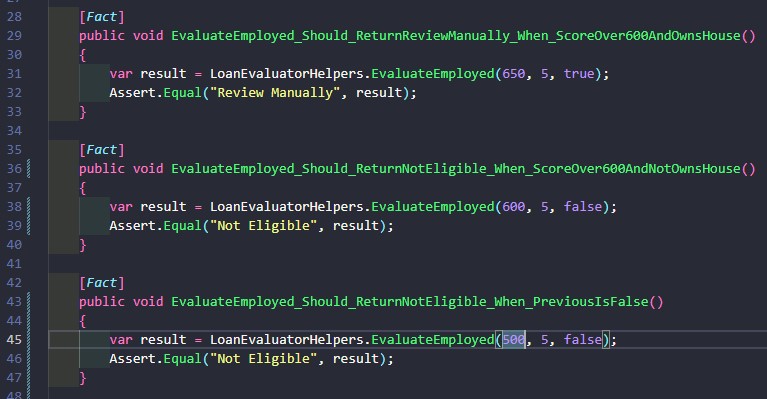




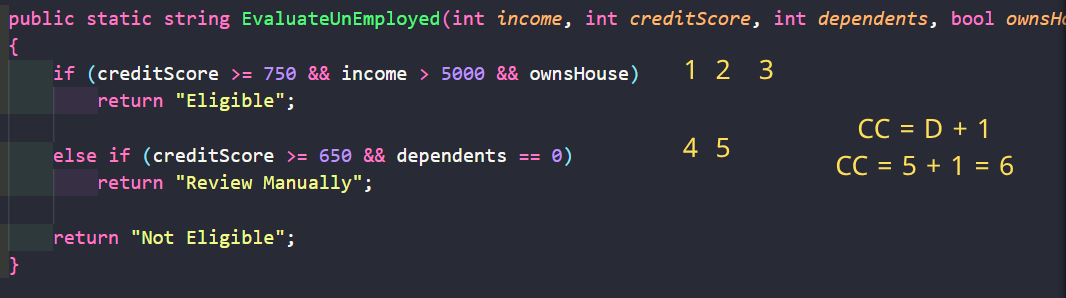


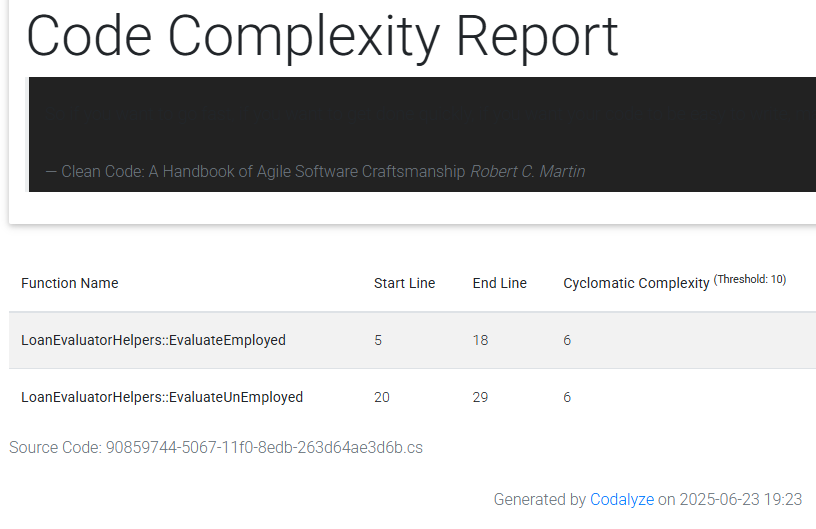
نلاحظ أن في التابع المساعد الأول EvaluateEmploye قيمة CC = 6 وبالتالي نحتاج على الأقل 6 توابع اختبار وهي:



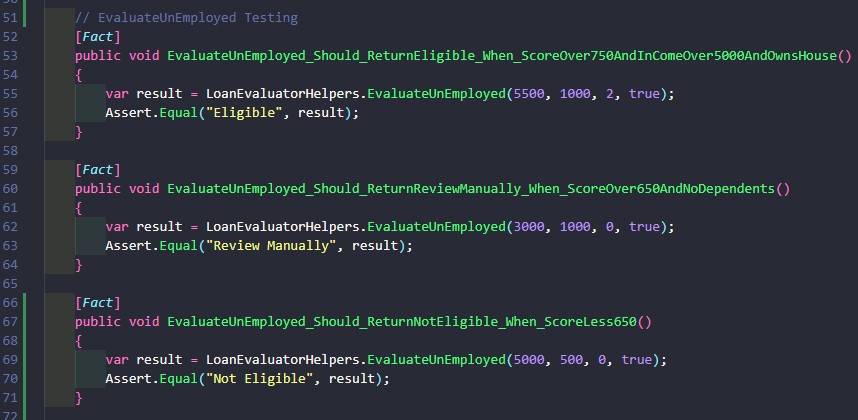


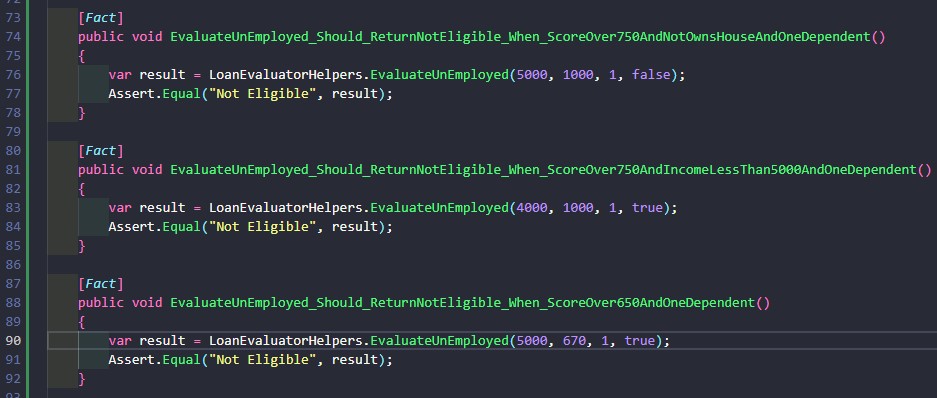
أما في التابع المساعد الثاني EvaluateUnEmploye:





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





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

