# CTEC 2902 Code Review Form

The following exercise accounts for 10% of the marks for the second half of CTEC2902. The deadline is the same as the final project deadline – 23:59:59 on the 3rd May.

## 1: Clean Code

Take one piece of code that you think is a *good example of clean code* from any of the code bases of the team projects in the CTEC 2902 organisation GitHub (note, you may need to switch from the main branches to development branches to find the code). This does NOT include the code for the labs.

The code should be as clean as possible, but also be code that does something complicated / significant. (It’s really easy to keep simple code that’s not doing very much clean). You can pick one or more methods, or a whole class, or even a couple of classes that interact with each other.

It can be your own code, or someone else’s. Keep the creator anonymous, though – it doesn’t matter who wrote it, just that you think it’s clean, and can explain why.

Paste the code into the box below (don’t worry if it scrolls onto the next page or three):

|  |
| --- |
| public void ConfigureAuth(IAppBuilder app)  {  // Configure the db context, user manager and signin manager to use a single instance per request  app.CreatePerOwinContext(ApplicationDbContext.Create);  app.CreatePerOwinContext<ApplicationUserManager>(ApplicationUserManager.Create);  app.CreatePerOwinContext<ApplicationSignInManager>(ApplicationSignInManager.Create);  // Enable the application to use a cookie to store information for the signed in user  // and to use a cookie to temporarily store information about a user logging in with a third party login provider  // Configure the sign in cookie  app.UseCookieAuthentication(new CookieAuthenticationOptions  {  AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,  LoginPath = new PathString("/Account/Login"),  Provider = new CookieAuthenticationProvider  {  // Enables the application to validate the security stamp when the user logs in.  // This is a security feature which is used when you change a password or add an external login to your account.  OnValidateIdentity = SecurityStampValidator.OnValidateIdentity<ApplicationUserManager, ApplicationUser>(  validateInterval: TimeSpan.FromMinutes(30),  regenerateIdentity: (manager, user) => user.GenerateUserIdentityAsync(manager))  }  }); |

Now explain in no more than one side of A4 (at the very most) why you think it’s a good piece of clean code. (You may need to refer to the *List of Code Smells* at the end of this document for more ideas about clean versus smelly code).

**What makes this a good piece of code**

This piece of code works well and it is clearly written out. By just looking at the code it is easy to understand what its purpose is for. The code contains a function which makes the users experience more efficient by using cookies. The code also contains some validation techniques for security purposes.

When a user logs into their account the system temporarily saves the data that is entered. The code uses a cookie function to do this. app.UseCookieAuthentication

This code also contains a security feature where the login details are validated within a given time of 30 minutes. OnValidateIdentity = SecurityStampValidator.OnValidateIdentity<ApplicationUserManager, ApplicationUser>(

validateInterval: TimeSpan.FromMinutes(30),

The layout of the code and the in line comments means that anyone can understand the code. The comments are clearly written and the naming conventions are used appropriately, the code also uses appropriate variable names.