# General

* .ConfigureAwait(false) Is not actually needed in the ASP.NET core as there is no SynchronisationContext and it does not have any effect in this solution.

# HomeController.cs

* It is not ok that Privacy endpoint is synchronously calling a service and blocking the calling thread.
  + Proposed solution: await should be used when calling \_privacyDataService.GetPrivacyDataAsync()

# StatisticMiddleware.cs

* InvokeAsync: The whole idea to create a task, attach a continuation to it and then wait (block the calling thread!) a hardcoded amount of time is bad. It would even stop working if the time required to register the visit increases and exceeds the hardcoded Thread.Sleep(3000)
  + Proposed solution: use await when calling async methods of \_statisticService
* It is better to register visits count and load the statistics view asynchronously, not impacting the time required to load the main page.
  + The statistics could be a partial view which is loaded asynchronously. This of course requires modifying html and javascript part of the solution.

# ManualAssistant.cs

* RequestAssistanceAsync try part: Similarly to the StatisticMiddleware.InvokeAsync, Thread.Sleep(5000) used for synchronization is a very bad idea.
  + Proposed solution: use await when calling async methods of \_supportService
* RequestAssistanceAsync catch part: Unnecessary creation of a wrapping tasks when string could already be returned directly.
  + Proposed solution: directly return the error message.