# LOGICAL AND REVIEW CODE

# AND

# BUILD WEB APP TEST BASED ON USE STORY

LEMBAGA PEMJAMIN SIMPANAN

2023

GROUP SISTEM INFORMASI
DIVISI PENGEMBANGAN APLIKASI

**Goals**:

This text is intended to measure the ability to write coding as well as measure the ability to carry out coding reviews.

Another technical measurement the candidate should be proven in web application development using clean architecture pattern.

**Task**:

- Please re-write the code, doing coding, analysing and fixing the code! (question number 1 to 7). Crate a solution using Microsoft Visual Studio and store all source code to github.com and create readme.md to explain your opinion!
- For number 8 read user story carefully and doing the task, and do not forget to store the code in github.com

## A. Logical and Review Code

1. How about your opinion..?

```
if (application != null)
{
   if (application.protected != null)
   {
      return application.protected.shieldLastRun;
   }
}
```

**Key**:
cleaner and easier to read code.

2. How about your opinion.

```
public ApplicationInfo GetInfo()
{
   var application = new ApplicationInfo
   {
      Path = "C:/apps/",
      Name = "Shield.exe"
   };
   return application;
}
```

**Key**:
return more than one value from a class method.

3. How about your opinion.

```
class Laptop
{
        public string Os{ get; set; } // can be modified
        public Laptop(string os)
        {
                Os= os;
        }
}
var laptop = new Laptop("macOs");
Console.WriteLine(Laptop.Os); // Laptop os: macOs
```

**Key**:
modifications by using private members.

4. How about your opinion?

```
using System;
using System.Collections.Generic;

namespace MemoryLeakExample
{
    class Program
    {
        static void Main(string[] args)
        {
            var myList = new List();
            while (true)
            {
                // populate list with 1000 integers
                for (int i = 0; i < 1000; i++)
                {
                    myList.Add(new Product(Guid.NewGuid().ToString(), i));
                }
                // do something with the list object
                Console.WriteLine(myList.Count);
            }
        }
    }

    class Product
    {
        public Product(string sku, decimal price)
        {
            SKU = sku;
            Price = price;
        }

        public string SKU { get; set; }
        public decimal Price { get; set; }
    }
}
```

**Key**:
Keeping references to objects unnecessarily

5. How about your opinion?

```
using System;
namespace MemoryLeakExample
{
    class Program
    {
        static void Main(string[] args)
        {
            var publisher = new EventPublisher();

            while (true)
            {
                var subscriber = new EventSubscriber(publisher);
                // do something with the publisher and subscriber objects
            }
        }

        class EventPublisher
        {
            public event EventHandler MyEvent;

            public void RaiseEvent()
            {
                MyEvent?.Invoke(this, EventArgs.Empty);
            }
        }

    class EventSubscriber
       {
            public EventSubscriber(EventPublisher publisher)
            {
                publisher.MyEvent += OnMyEvent;
            }

            private void OnMyEvent(object sender, EventArgs e)
            {
                Console.WriteLine("MyEvent raised");
            }
        }
    }
}
```

**Key**:
event handlers

6. How about your opinion?

```
using System;
using System.Collections.Generic;
namespace MemoryLeakExample
{
    class Program
    {
        static void Main(string[] args)
        {
            var rootNode = new TreeNode();
            while (true)
            {
                // create a new subtree of 10000 nodes
                var newNode = new TreeNode();
                for (int i = 0; i < 10000; i++)
                {
                    var childNode = new TreeNode();
                    newNode.AddChild(childNode);
                }
                rootNode.AddChild(newNode);
            }
        }
    }

    class TreeNode
    {
        private readonly List<TreeNode> _children = new List<TreeNode>();
      public void AddChild(TreeNode child)
      {
            _children.Add(child);
        }
    }
}
```

**Key**:
Large object graphs

7. How about your opinion?

```
using System;
using System.Collections.Generic;
class Cache
{
  private static Dictionary<int, object> _cache = new Dictionary<int,
object>();

  public static void Add(int key, object value)
  {
    _cache.Add(key, value);
  }

  public static object Get(int key)
  {
    return _cache[key];
  }
}

class Program
{
  static void Main(string[] args)
  {
    for (int i = 0; i < 1000000; i++)
    {
      Cache.Add(i, new object());
    }

    Console.WriteLine("Cache populated");

    Console.ReadLine();
  }
}
```

**Key**:
Improper caching

## B. Web Application Development using Clean Architecture

No 1. Read **User Story** below carefully:

The business unit require a web application to receive document (xlsx and pdf) from customer and store those documents into database. Before send the required documents customer should be register if not registered yet. System should be authenticated the user as customer before send documents to the system. In Password should be fulfilled with requirements as stated below:

- It contains at least one lowercase English character.
- It contains at least one uppercase English character.
- It contains at least one special character. The special characters are: !@#$%^&*()-+
- Its length is at least 8.
- It contains at least one digit.

When documents store successfully in database (transaction completely), system send notification to customer as receipt that document is submitted successfully. The size of Documents are more than 1 or 2 GBs, and technically required handling like chunking method.

The business unit able to monitor the documents which sent by customers and unit business able to download those documents.

**Task**:

Develop web application using clean architecture web API .net core 7, and entity framework using c#.