# Coding test

We want to expose a service that calculates the total spend given a supplier ID.

public class SpendService

{

public SpendSummary GetTotalSpend(int supplierId) { ... }

}

The business logic is quite straightforward: the total spend is the sum of all the invoices, grouped by year. However, some of the suppliers are working with a separate branch of the company, which has its own invoicing platform.

Therefore, the flow should work as follows:

1. A **SupplierService** returns supplier data, which can be used to understand whether a supplier is external or not
2. If the supplier is **not external**, invoice data can be retrieved through the **InvoiceRepository** class
3. If the supplier is external, invoice data can be retrieved through the **ExternalInvoiceService** class
4. **ExternalInvoiceService** invokes a separate system, which might fail. However, data from this system is regularly backed up in a failover storage. A **FailoverInvoiceService** class gives access to that storage. It is ok to return failover data when **ExternalInvoiceService** fails.
5. Failover data might be not fresh. A timestamp property indicates when it has been originally stored. If this date is older than a month, it means that it has not been refreshed. In this case, the **GetTotalSpend** method should fail.
6. When ExternalInvoiceService is offline, usually calls tend to **timeout**, which means that the method takes long to complete. Therefore, after 3 consecutive errors, we want to **bypass** ExternalInvoiceService and go to FailoverInvoiceService **directly**, with the same logic as before. After 1 minute, we can try to re-enable ExternalInvoiceService again.

## Rules

You can use any framework(s) of your choice.

You can make changes to the following classes:

* SupplierService
* InvoiceRepository
* FailoverInvoiceService

However, you can’t modify the signature of any methods.

Classes in the **ProArch.CodingExercise.External** library cannot be modified – it’s considered as an external SDK we don’t have any control over.

## Goal:

Implement the GetTotalSpend method and all the unit tests you believe are worth implementing. When you do it, you should consider the following: SOLID principles, maintainability, testing.

**The finalised solution must build and all the tests must pass, we won’t look at code that doesn’t meet these criteria.**