1. **Introduction**

MarkMonitor aims to maintain a stimulating work place with dedicated professionals, where everyone understands the importance of high quality, maintainable code, and scalable solutions. The tasks given to candidates being considered for open positions are meant to ensure that the developers we hire are well suited for the work in our team. Because we are doing this, we can guarantee a very interesting and vibrant work environment, with smart, experienced and highly motivated people, who learn from each other and develop enterprise-grade solutions together, using the newest platforms and technology available.

We very much appreciate your interest in the company, and the time you spend on this. There are three parts to this task you will need to complete:

1. **Web Crawler Implementation** – deliver a working Visual Studio solution
2. **Code Review** – write in-line response in this document
3. **Process Notes** – write in-line response in this document

When preparing your solution, keep in mind that it will be a significant element in our evaluation of you as a candidate for the position, and that the submitted material will undergo a thorough internal review, and will be checked for plagiarism. This is meant to show us your abilities and approach to a development task.

1. **Web Crawler Implementation**

*Develop a simple Web Crawler application using Visual Studio 2010 and C#/.NET 4.0*

The application should be able to:

* Download a given webpage from a website
* Using appropriate techniques, extract the hyperlinks found on the downloaded page
* Store the links in a database
* Fetch new links from the database and display in a UI
* Continue to crawl the new links found

Notes:

* Use multithreading and event handling where it is feasible
* The application must compile and run in Visual Studio 2010 or 2012 (must include the data store added to the project, as well as all necessary libraries and resources)
* As a guideline, you should spend maximum 8 hours in total to develop the application

If you use any components you did not write yourself, other than .NET, please clearly state where and why it was used in the **Process Notes**. Hand in a solution you are pleased with, but also be sure to explain where and how it could be improved in the **Code Review**.

A copy of your solution should be provided in a RAR or ZIP file, together with a copy of this document, with the following sections filled in with your responses.

1. **Code Review**

*Provide a review of your code, including notes about any problems or weaknesses present in the solution, and describe what you would improve in future iterations (max 1 page):*

***Code review Initial thoughts***

*The solution presented here has been made over several iterations, heavily interrupted by ‘family duties’, during my Christmas holly days. It has taken more than the suggested 8 hours. Does this reveal inability to limit the scope? Maybe! I have had lots of fun during this period. I’ve been caught by interesting problems that I wanted to investigate.*

*The solution might seem over engineered, but on the other hand it allows me to demonstrate my knowledge about several different fields of technology.*

***What has been presented?***

1. *An isolated, testable domain model residing in the WebCrawler.Domain project.*
2. *Use of an existing framework Sharp architecture, displaying knowledge about DDD concepts as Entities, value objects, etc. (In particular notice the use of [DomainSignature] to get automatic working Equality computation.*
3. *Use of an ORM framework (NHibernate)*
4. *Use Of Command, Query Seperation*
5. *Use of ASP.NET MVC 4 framework.*
6. *Use of design patterns: layered architecture, Seperations of concerns, Command, Factory, Repository, Single Responsibility Principle, Query, lazyLoad, + more*
7. *Use of an IOC container(Castle Windsor)*
8. *Use of BBD driven design with MSpec.*
9. *Automatic Database Schema generation (WebCrawler.Tests project, CanCreateDatabase)*

***Alternatives and improvements?***

1. *No one says I should make a web application*
2. *Events could have been used more. Domain events, events from the Page class.*
3. *More use of new async pattern in .net 4.5 would yield higher/a degree of parallelism.*
4. *Error handling needs to be fully implemented.*
5. *Use of T4 templates to generate repetitive code like Commands, ViewModels, Queries, CommandHandlers.*
6. *Browsing the whole internet leads the mind to technologies like BIG DATA, NoSQL, Hadoop and MapReduce.*
7. *A more minimalistic approach, if the task was: “write a web crawler using as few lines of codes as possible”.*
8. *Demonstration of persisting to the database is only partly demonstrated in the website project. An improvement could be to raise an BatchSizeHit Event. Some code responsible for persistense could invoke an Update CrawlerSession Command Handler.*
9. *The implementation will create a big object graph, which could be a problem. An alternative solution is to make a relation table in memory; sometimes referred to as an ‘Adjacency Matrix’*

***Code style***

1. *I’m influenced by authors like Robert.C.Martin (Clean Code), Craig Larman (Applying UML and Patterns), Abrams, B. & Cwalina K (Framework design guidelines), and also E.Evans (The DDD book) + others. So I try to adhere to SOLID principles like Single Responsibility, DRY etc. I also use GRASP patterns presented by Craig Larman. Regarding code guide lines, my spin on that matter is that now a day it’s easy to refactor your code to follow some proprietary code guideline using Resharper. And regarding comments Robert.C.Martin puts it nicely: Adding a comment should not be an excuse for writing sloppy code. Rahter than focusing on having all methods commented, I find it far more important that the classes and methods are small, simple and testable,. (An API should Be nicely commented though) This is secured by using inversion of control principle, and by using TDD approach. So any short comings in my example like missing comments broken naming conventions, can be fixed with Resharper, like mentioned above.*

1. **Process Notes**

*Explain how you approached the development task, and describe the process of solving it (max 1 page):*

1. Googled “C# Webcrawler”. To get some inspiration and ideas. First hit let me to <http://www.thecodinghumanist.com/Content/HowToWriteAWebCrawlerInCSharp.aspx> I conclude that the point here is to use the HttpWebRequest and the WebResponse class to do the low level job.
2. I think about the task. Important points are
   1. It’s a graph problem where links are paths and webpages are the nodes.
   2. Search depth
   3. Recursiveness
   4. Cyclic links
   5. Max amount of memory to use
   6. When to save a batch of webpages.
3. I think about what object I will start out with. My candidates are:
   1. A Link object, with source and destination, I decide it must be a value object. Uniquely determined by source url and destination url.
   2. A webpage object with a unique address. Maybe it only has a relative address. I decide to treat it as an entity until further. It will also not be immutable.
   3. I think I want a CrawlerSession, and maybe a crawler settings object also. The CrawlerSession can also be a AggregateRoot.
4. What approach should I take? Hmm. Since DDD is mentioned in the job add, and since its one of my main reasons for applying for the job, I will try to show my skills in that field. I Will Use the Sharp Architecture Framework. <http://sharparchitecture.net/>
5. So let’s get started.
   1. I Create a new visual studio solution based on Sharp Architeture.
   2. I port it to visual studio 2012
      1. I Replace asp.net mvc project with a new asp.net mvc4 project
      2. I update .net framework to 4.5 on all projects
   3. I Add a class “LinkSpec” to MSpecTests.Webcrawler project
      1. I Create a specification for constructing a link
   4. I Add the Link class to WebCrawler.Domain project
   5. I Make tests build
   6. Resharper doesn’t recognize MSpecTests, I google and find a solution <http://awkwardcoder.blogspot.dk/2012/06/getting-mspec-working-with-visual.html>

Upgrade MachineSpecification nuget package to latest beta, and run batfile from mspec nuget package folder.

* 1. Now tests run

