1 Introduction:

This document contains the technical details regarding the deliverables mentioned in the evaluation test. This document will consist of a few screenshots of the solution and the workflow adopted to achieve it. The solution was based on extracting doxygen style comments from a cpp source file and outputting them in xml file format. This would then be fed as input to doxybook2 in the command prompt and the xml files would be transformed into markdown files to be used as content in any static site generator which in this case is Hugo.

2 Setting up the solution:

The first step is to install doxygen on your system which can be done by visiting their website at doxygen.nl. For windows, we navigate to the downloads tab and select the respective binary to download it. After downloading we run the executable file and follow the installation steps listed. Doxywizard will be available after the installation of doxygen is complete. We can configure many of the important settings in the Doxywizard GUI frontend as can be seen in Figure 1 below:

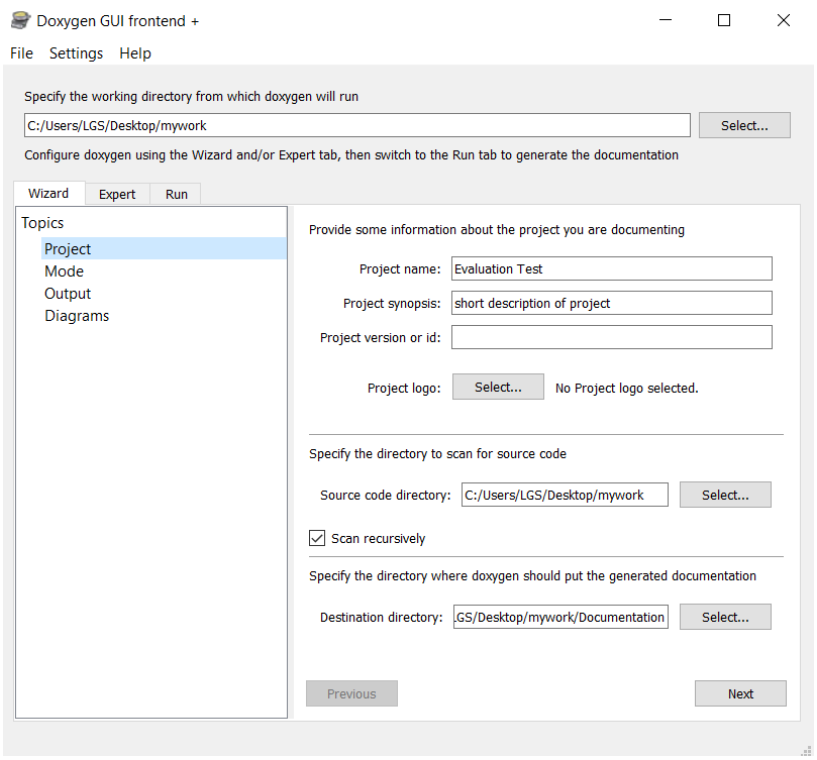

Figure 1 Doxywizard GUI Frontend

The next step is to install the doxybook2 executable file onto our system and add it to our PATH. Doxybook2 runs in the command prompt and can also be used as a C++ library. After installing the precompiled binary listed in the releases of the github repository, we must add it to the user variable: PATH so that it can be detected by our command prompt. We can check whether it has successfully been added to the path by typing “doxybook2 -h or --version” which will show us the options menu as follows:

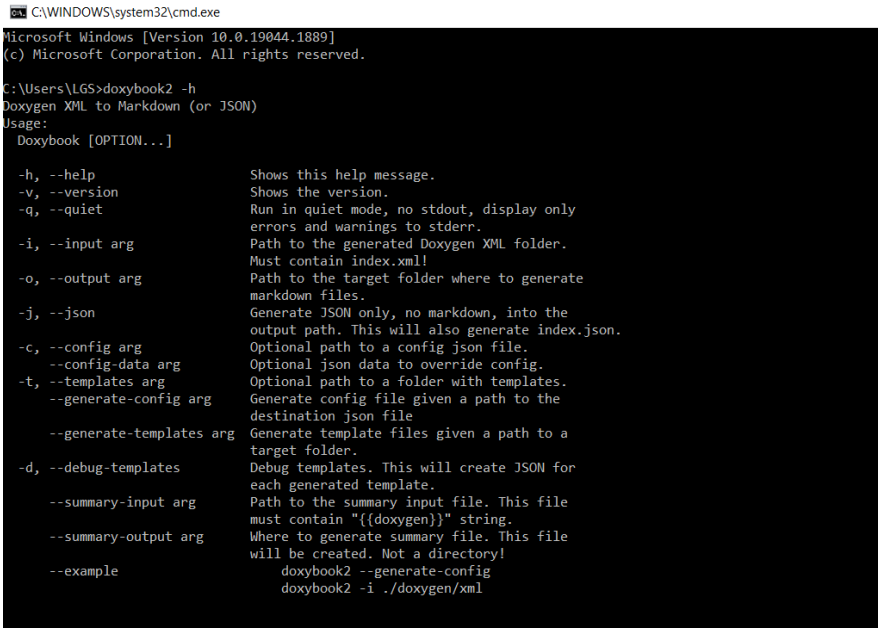

Figure 2 Doxybook2 Command Prompt

The last component to install is the hugo static website generator. We can go to gohugo.io and navigate to the release page. There we can see the latest releases and we can install the zip file based on the OS that we are using. After extraction into our new directory, we then have to add it to our user variable: PATH. After this is done, we can start creating our website through the command prompt using a simple “hugo new site website_name”. We can continue with the configuration afterwards.

3 Using the solution:

The tasks in this evaluation test are divided into three parts. The first is to extract XML files from a cpp source code file using doxygen. The cpp file must include doxygen style comments so that the doxygen software can read and extract them accordingly. The doxywizard has many different options for outputting the extracted comments. For this case, we will be choosing XML output since we will be using doxybook2 to convert these XML files into markdown files. For the doxywizard settings, we will be setting our output to extract from all the files in the source directory which contain any doxygen comments. After configuring the doxywizard, we can see the xml files being generated in our specified directory folder and can be seen as below:

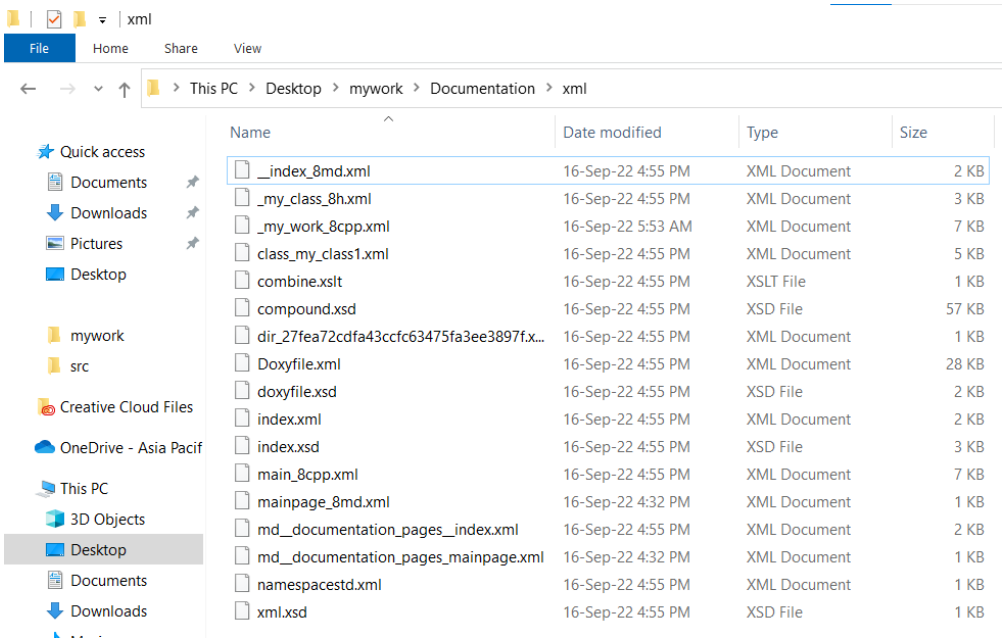

Figure 3 XML files extracted using doxygen

After these files have been extracted, we will run doxybook2 in the command prompt and enter the following line: “doxybook2 --input (path) --output (path) --config (path to config.json)”. The description for each is as follows:

- -- input is the input path which guides doxyboo2 to the folder where the XML files were extracted.
- -- output is the destination path as to where we want our converted markdown files to reside.
- -- config is the file which contains all the configuration settings regarding the XML files and how to feed them to a static website generator like Hugo.

We can see the doxybook2 command in the CLI as follows:

```
C:\Users\LGS>cd C:\Users\LGS\Desktop\mywork

C:\Users\LGS\Desktop\mywork>doxybook2 --input C:\LGS\Desktop\mywork\Documentation\xml --output C:\Hugo\Sites\test.com\content --config "C:\Users\LGS\Desktop\mywork\config.json"
```

Figure 4 Doxybook2 CLI command

Using the command in Figure 4, we can see the newly converted markdown files in our output directory and can be seen in below:

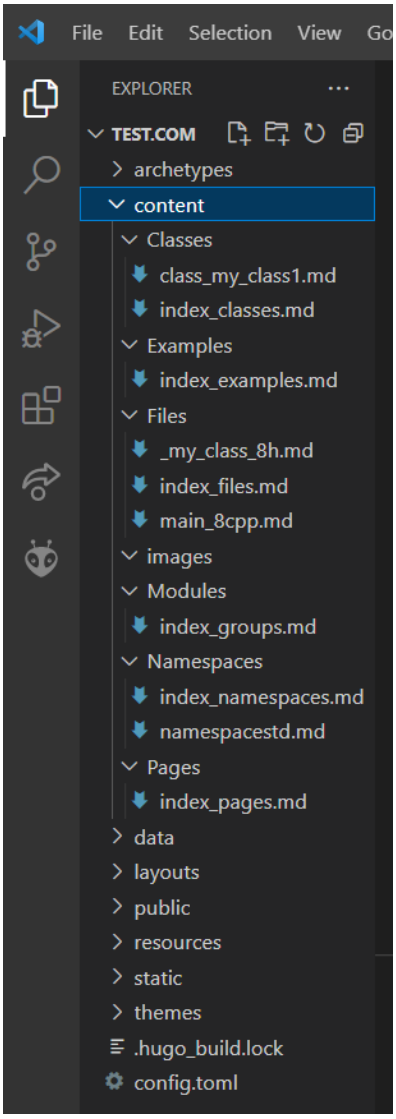

Figure 5 Markdown files

Since we have put these markdown files in the content directory of Hugo, this means that we only need to run a Hugo server and the files will be displayed on the url stated: localhost:1313. However, for this case, the markdown files were not being read properly in the content folder of Hugo and thus the screenshots added now will be for the Doxygen HTML output instead of the Hugo static website.

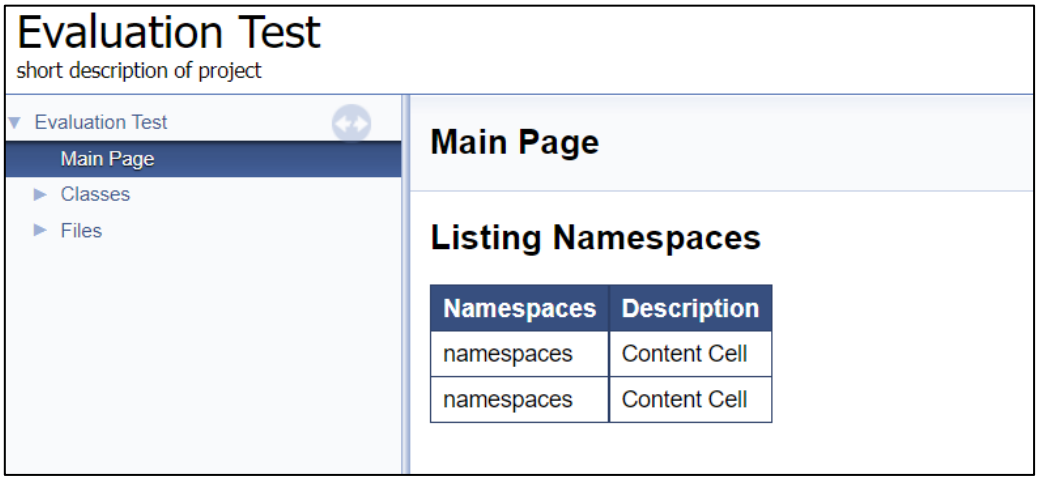

Figure 6 Main Page with Namespaces

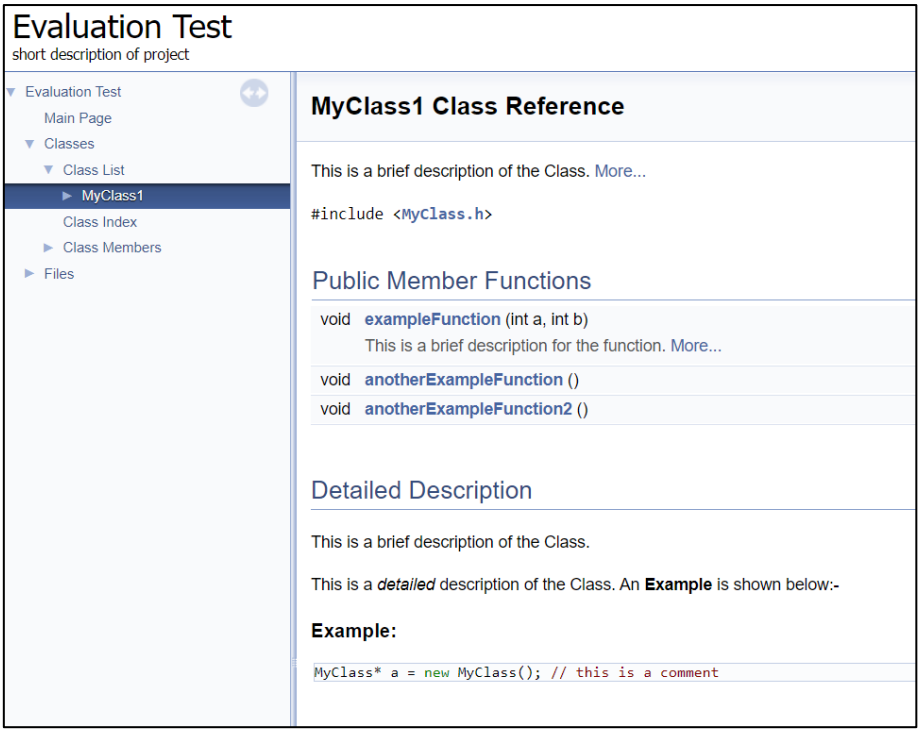

Figure 7 MyClass1 in Doxygen HTML

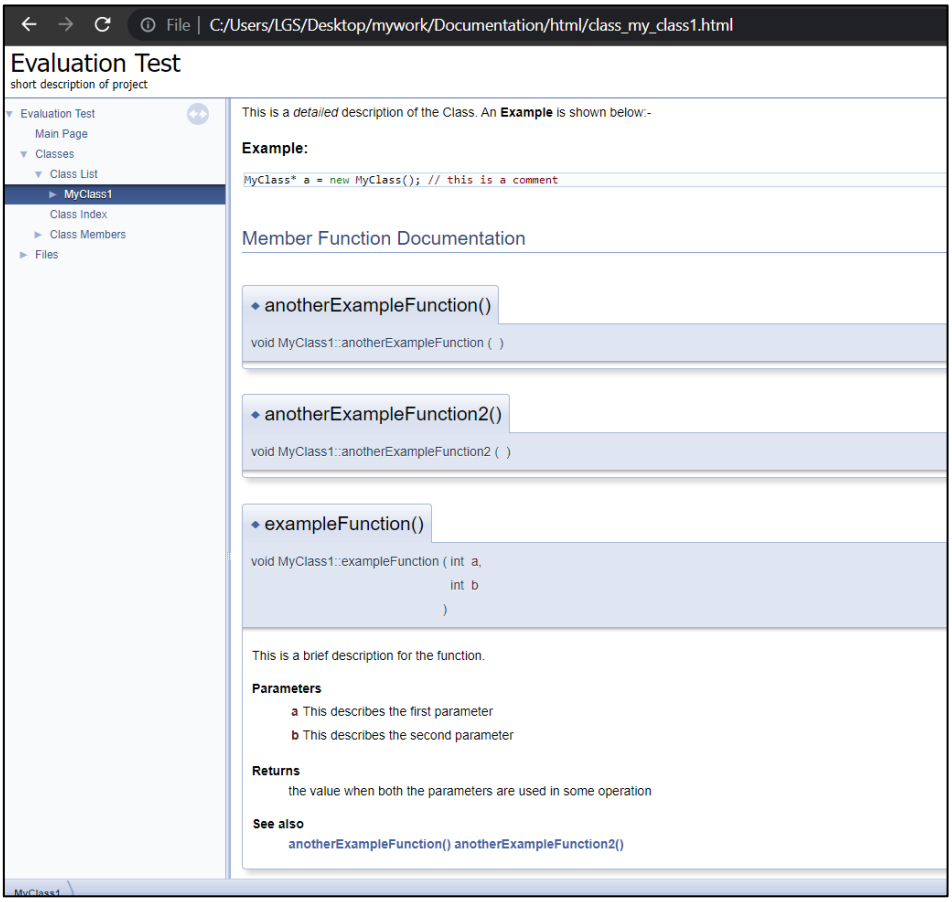

Figure 8 Public Member Functions in Doxygen

4 Methodologies and Approaches:

The methodologies and approaches used in this task were based on the XML to MD conversion using XSLT which is a method to code XML files. In this case, Doxygen software converted the specially coded comments into XML through its inbuilt code. The markdown files that were created for the mainpage used extended syntax to some point whereas the rest was simply converted through the doxybook2 command line execution. Since I had no prior experience in this field, there was a lot of research and then after many trial and errors, a certain percentage of the evaluation test was successfully completed. There is great room for improvement in the given attempt and could’ve been a much better deliverable with better knowledge on the subject matter.