XMLPad

Empower your text editing with XMLPad - the ultimate XML editor.

Final report

Submitted for the BSc in

Computer Science(Software Engineering)

April 2023

by

Maykal Mariyanov Droshev

Word count: ***XXXXX***

Abstract

XMLPad is a text editor that allows users to modify, create and compare all popular text file formats. Designed for students and academics, XMLPad fills the gap of missing WPF-open source text editor that has most of the capabilities of Notepad++ and some of the functionality of Microsoft Visual Studio. With the emerging popularity of “Dark mode”, XMLPad excels in delivering it alongside other features, including text comparison, finding and replacing text, formatting and Tabifying XML and general programming code. Originally designed as a basic text editor with the editing features like cut, paste, and save, XMLPad included one of the most popular and complex services, whilst remaining robust and performant. XMLPad allows the user to change the colour scheme of the editor, depending on the programming or markup language that is being edited. The main standing point of the software is its educational purpose for enthusiasts that want to learn WPF and the algorithms behind one of the most popular editing functionalities. Alongside this, academics and software developers can alter the software, depending on their specific needs. With XMLPad, editing and learning how it's done has never been easier.

Acknowledgements

I desire to acknowledge several individuals and organisations which provided beneficial support to this project.

To start with, thanks to Daniel Grunwald for sharing the code and implementation of [AvalonEdit](http://avalonedit.net/). AvalonEdit is the heart of the project and the text editing element on the [main window](#_7.2.2__Main) of the software. With his code, I was able to implement line numbers and built in functionalities in way shorter period of time than what was planned initially. That allowed more time for additional implementation and testing of new code.

There is a possibility that our software developing jobs will be taken away from us one day by this organisation. However, I would like to acknowledge OpenAI and their [ChatGPT](https://openai.com/blog/chatgpt) for explaining complex code, concepts and providing snippets that optimise the overall performance of the software in a matter of seconds (Unless the server is at maximum capacity).

I would also like to give special thanks to Neil Fraser for providing the community with his implementation of diff-match-patch – code that takes two pieces of text and returns the differences from both of them. His code is mainly installed into the [*Compare Text*](#_7.5.1_Compare_Window) window.

Many thanks for the guidance and support from Dr. Septavera Sharvia, through the entire project. The advice and mentorship received will always be remembered.

Without these people and organisations help, the work on this project would be nearly impossible to be done in the recommended timeframe.

Contents

[Abstract i](#_Toc129868720)

[Acknowledgements ii](#_Toc129868721)

[1 Introduction 3](#_Toc129868722)

[1.1 Background to the project 3](#_Toc129868723)

[1.2 Aims and objectives 3](#_Toc129868724)

[1.3 Research question 3](#_Toc129868725)

[2 Literature review 4](#_Toc129868726)

[3 Requirements 5](#_Toc129868727)

[3.1 Product requirements 5](#_Toc129868728)

[3.2 Functional requirements 5](#_Toc129868729)

[3.2.1 Interfaces 5](#_Toc129868730)

[3.2.2 Functional Capabilities 5](#_Toc129868731)

[3.2.3 Performance Levels 5](#_Toc129868732)

[3.2.4 Data Structures/Elements 5](#_Toc129868733)

[3.2.5 Safety 5](#_Toc129868734)

[3.2.6 Reliability 5](#_Toc129868735)

[3.2.7 Security/Privacy 5](#_Toc129868736)

[3.2.8 Quality 5](#_Toc129868737)

[3.2.9 Constraints and Limitations 5](#_Toc129868738)

[3.2.10 Performance requirements 5](#_Toc129868739)

[3.3 Design constraints 5](#_Toc129868740)

[4 Design 6](#_Toc129868741)

[4.1 Software design 6](#_Toc129868742)

[4.2 Hardware design 6](#_Toc129868743)

[4.3 Experimental design 6](#_Toc129868744)

[5 Implementation and testing 7](#_Toc129868745)

[5.1 Implementation 7](#_Toc129868746)

[5.2 Testing 7](#_Toc129868747)

[6 Evaluation and discussion of results 8](#_Toc129868748)

[7 Conclusion 9](#_Toc129868749)

[References 10](#_Toc129868750)

[Appendix A – Interesting but not vital material 11](#_Toc129868751)

[Appendix B – Images 12](#_Toc129868752)

[7.1 Windows 12](#_Toc129868753)

[7.1.1 Welcome Window – Light Mode 12](#_Toc129868754)

[7.1.2 Welcome Window – Dark Mode 12](#_Toc129868755)

[7.2.1 Main Window – Light Mode 12](#_Toc129868756)

[7.2.2 Main Window – Dark Mode 12](#_Toc129868757)

[7.3.1 Find/Replace Window – Light Mode 12](#_Toc129868761)

[7.3.2 Find/Replace Window – Dark Mode 12](#_Toc129868762)

[7.4.1 Manual – Dark(Always Dark) 11](#_Toc129868763)

[7.5.1 Compare Window – Light (Always Light) 11](#_Toc129868764)

# Introduction

The project consists of a software called XMLPad. The software is open-source, WPF based text editor which successfully edits XML alongside some of the most pupular programming and markup languages. The major stakeholders in the project are the users, represented in students, academics and curious developers. Next to them are the developer of the project, alongside the project supervisors, Dr Septavera Sharvia and Dr Baseer Ahmad.

The report structure will be mainly described in the [Contents](#_top) table and it will follow chronological order starting from the submission of the Project Initiation Document.

The project as a whole, also consists of the PID, this report, meetings with supervisors, video that showcases the software major key points and future plans, and a demo of the software with the supervisors to discuss how the project went and any commenting on the overall performance.

## Background to the project

## Aims and objectives

## Research question

# Literature review

If we start with the foundation of the project, WPF. Windows Presentation Foundation is graphical subsystem for Windows based applications. Developed from Microsoft for rendering user interfaces windows, WPF is used for developing Visual Studio, Microsoft Expression Blend, JetBrains Raider, Discord and many more popular software products. WPF is combination of XAML or Extensible Application Markup Language for the visual interface and C# for the business logic. Despite the general mass using web applications and native mobile apps, WPF is developed with powerful tools and massive library of resources, tutorials and libraries.

Mentioning libraries, In the core and heart of the project software stays AvalonEdit. AvalonEdit is WPF-based RichTextBox component, developed by SharpDevelop/ Daniel Grunwald that includes a wide set of features, which can be seen in the project. Although, the original RichTextBox is powerful for general purpose WPF apps, it can prove quite limiting when we try to implement syntax highlighting or code folding, or other features. Some of them include token based colouring, line numbering, folding of code blocks, auto complete, search and replace. It also integrates well with other tools such as spell checkers and diff tools.

Another library used in XMLPad is diff-match-patch. The code in it, gives the ability to compare and manipulate plain text. Provided that the developer copies the open-source code from [gitHub](https://github.com/google/diff-match-patch/blob/master/csharp/DiffMatchPatch.cs), we can have a full window that has the differences between two files in less than four lines of code. Under the repository, developed by Google, diff-match-patch has implementation across all popular programming languages.

# Requirements

The main requirements for the project were a graphical editor tool that includes features like cut, paste, save, etc. In the Project Initiation Document, more objectives were created to make to project stand out and excel. Whilst developing the software, more features were included as the timeframe was allowing for it. The next sections will be separated between Initial Requirements and Development Requirements to examine the actual progress of the project software.

## Product requirements

What will your software or hardware do? Who requires it? You might want to refer back to your aims and objectives to inform this section, and perhaps consider if they are still appropriate. UML use case diagrams are very helpful here (even for hardware).

Delete the red paragraph and replace this one with your content (use the “Normal” paragraph style).

## Functional requirements

The exact content here will vary (especially if your project is hardware-based), but there are some standard items which you should consider including:

### Interfaces

Initial Requirement:

Graphical user interface program that is easy to navigate and use. The software should be appealing to the eye and robust enough to sustain a smooth and seamless user experience. The software should allow for dark and light mode and have code colouring for XML.

Development Requirement:

Dark and Light theme were implemented across almost all windows and code colouring was implemented for all of the most popular languages, including Boo, Coco, C++, C#,CSS, HTML, Java, etc.(14 in total). On top of that, code colouring is changed if we change the theme of the app. Also, code colouring is saved when the user closes the app. For example, if we develop a snippet in C# and dark mode and we close the app, the next time these two settings will be preloaded already. Almost all of the visual elements are dynamically changed to achieve aesthetic looks and professional design.

### Functional Capabilities

### Performance Levels

### Data Structures/Elements

### Safety

### Reliability

### Security/Privacy

### Quality of Code

No predefined code quality requirements. On development stage many standards were implemented, including code separation, code comments, file and folder structuring for different windows, images and resources. GitHub repository was kept up to date with small working changes one at a time, resulting in more than seventy commits. Readme file was created, which helped with learning about markdown language and how to use it. A big progress that shows how well created was XMLPad is the fact that the ReadMe file was created on the XMLPad itself. MIT License comments were implemented.

### Constraints and Limitations

The three main constraints for any sort and type of project are money, time and quality. As this is a project that will not be commercialised, it doesn’t involve other people and it doesn’t require any other equipment than a laptop, money cannot be considered in the equation. We have time and quality. In the absence of money, these two are really tightly coupled. This was identified on the Project Initiation document phase and huge efforts were made to make sure enough time is given to the project so it can live up to the expectations and exceed them. The quality of the software is also affected hugely from one major factor – experience in software development and Windows Presentation Foundation. Given the fact that only one lecture was given on WPF (first year, second trimester, as part of Dr John Dixon lectures from Object Oriented Programming) additional courses were taken in order to get more knowledge and experience.

### Performance requirements

## Design constraints

You might include this in the next section if you prefer. Consider the limitations on how you are able to conduct your project. Relate the bounds (time and resources are obvious ones) which have an impact.

Delete the red paragraph and replace this one with your content (use the “Normal” paragraph style).

# Design

If your project involves the development of software and/or hardware, then you will need to include a section in which you describe its design in detail. If you conduct any experiments (either in a research-oriented project or simply doing user evaluation) then you should describe their design and methodology here.

Delete the red paragraph and replace this one with your content (use the “Normal” paragraph style).

## Software design

Typical content will be detailed software design, from architecture to implementation level. As well as your text, you should include UML diagrams, including class structures, activity and sequence diagrams as appropriate. Don’t just drop diagrams in willy-nilly, though. Use them strategically to illustrate points in your text. Remember that ‘a picture is worth a thousand words’ (we don’t apply this rule literally) but pictures on their own don’t explain everything.

If your project requires user interface design, don’t forget to include that. Screenshots, wireframes and other diagrams are welcome.

Delete the red paragraph and replace this one with your content (use the “Normal” paragraph style).

## Hardware design

If your project involves building hardware, give full details about the process here. Include diagrams as appropriate Use them strategically to illustrate points in your text. Remember that ‘a picture is worth a thousand words’ (we don’t apply this rule literally) but pictures on their own don’t explain everything.

If your project requires electronics and/or mechanical design, don’t forget to include that. Photos, CAD drawings, electronic schematics and other diagrams are welcome.

Delete the red paragraph and replace this one with your content (use the “Normal” paragraph style).

## Experimental design

If you are going to evaluate your software or hardware by means of any tests or surveys, then explain their design here. If you are doing other experiments (for example measuring the performance of algorithms, extracting data from environmental monitoring systems or evaluating the performance of mechanisms) then you should explain how you have designed the experiments, how they must be conducted and what you expect to learn from them. This is especially important for research-oriented projects.

Delete the red paragraph and replace this one with your content (use the “Normal” paragraph style).

# Implementation and testing

Delete the red paragraphs and replace this one with your content (use the “Normal” paragraph style).

## Implementation

In this section you will describe what you did, and why you made the important decisions affecting your actions. It’s not a diary – don’t write a blow-by-blow account of every little thing that happened. Be selective and report those choices and techniques which made a difference. Make sure you discuss what options you considered. Explain how the criteria and methodology you used to select amongst different options (which tools are most appropriate, for example).

It may help to imagine that you are reading this project in the future, trying to replicate the work without making the same mistakes along the way. What would you need to know to make your job easier, and what is unimportant or obvious? Explain how you implemented the design in the previous chapter.

This is the place in which you would explain any novel or especially complex algorithms, data structures or systems you have used.

Make it clear what you have done, and what is pre-existing. For example, if you are using third party software libraries, describe how you have used them, and how they have benefited your project rather than simply what they do. If you have built on a framework, make it clear how you have developed new functionality.

Delete the red paragraphs and replace this one with your content (use the “Normal” paragraph style).

## Testing

If you are developing software or hardware, you must test it. This section should explain how your work will be (or has been) tested.

You should have a test plan at the very least (full details of it and its results if required can go in an appendix). Ideally, you will have automated tests for any software you build. You will also define user acceptance tests, or something similar which can be used to determine whether your output meets the requirements stated earlier. Explain how and when the tests should be conducted.

Delete the red paragraphs and replace this one with your content (use the “Normal” paragraph style).

# Evaluation and discussion of results

This section evaluates the *software (or other artefact)* you have developed. You should compare it with the original specification and see how well it satisfies the requirements. You may wish to refer back to your aims and objectives at this point. You should report the results of user testing and a summary of feedback if that has been collected.

If you have done experiments, then the results of these should be reported and discussed here.

If you have involved people in doing user evaluations, that information should be include here.

Delete the red paragraphs and replace this one with your content (use the “Normal” paragraph style).

# Conclusion

In this section you should evaluate the *project* as a whole, and draw conclusions from the work you have done. Ask yourself what the project has achieved – what is its contribution? Has it met its initial aims and objectives? If not, why? How does the work you have done enhance the field in general? What has been learned from the project? If you have a well defined research question, has it been answered? What do the results mean?

You should also use this section to reflect on the *process* by which you undertook the project. Was your methodology appropriate (and did you stick to it)? Was your time planning good? Did you complete the primary and secondary objectives, and if not then why? What have you learned from the process? What would you do better/differently if you had more time?

Sometimes, it’s appropriate to include a subsection on ‘Further work’, making suggestions of how to proceed and what could be done to enhance the project in future.

Delete the red paragraphs and replace this one with your content (use the “Normal” paragraph style).

References

References must be formatted in the correct manner. For this assignment you must use the University of Hull’ approved variant of the Harvard referencing style (Fallin 2019), fully described at https://libguides.hull.ac.uk/referencing/harvard. Note that the details of the expected format vary depending on the type of document being referenced. Make sure you are familiar with them. If you use reference management software such as Zotero, EndNote or RefWorks, then you should be able to export a table of references in the correct format, which will save you work.

Every reference should have at least one citation in the text. Most will probably be in the ‘Background’ or ‘Literature review’ sections.

Remember that there is a difference between references and a bibliography. You will certainly need references, but a bibliography is optional.

There is much more information and guidance about referencing on the library’s website at https://libguides.hull.ac.uk/referencing/home

Some examples, illustrating different types of source:

Bahraini, M.S., Bozorg, M., Rad, A.B., (2018). SLAM in dynamic environments via ML-RANSAC. *Mechatronics* 49, 105–118.

Fallin, L., (2019)*. LibGuides: Referencing your work: Harvard Hull.* Available online: http://libguides.hull.ac.uk/referencing/harvard (accessed 10/10/2019).

Janis, I., (1972). *Victims of Groupthink: A psychological study of foreign-policy decisions and fiascoes.* Houghton Mifflin, Boston.

Office For Students (2018) *. Securing student success: Regulation framework for higher education in England*. Available online: https://www.officeforstudents.org.uk/media/1406/ofs2018\_01.pdf (accessed 10/10/2019)

Schmuck, P., Chli, M., (2019). CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. *Journal of Field Robotics* 36, 763–781.

Delete the red paragraphs and replace this one with your content (use the “Normal” paragraph style).

Appendix A – Interesting but not vital material

Appendices are used to include information which may be of interest but is not necessary for the reader. You do not have to include appendices if there is no need for them.

You might, for example, want to include some details of a particular piece of software (an API, perhaps) or hardware which your project uses. This might be something that a reader might wish to consult, but you wouldn’t want to include in the main body of the report. You could also put raw data from experiments in an appendix, or perhaps survey results. It should still be information of relevance, but nothing that everyone would be expected to read.

If you wish to refer to elements of your PID, you could include them in appendices.

Delete the red paragraphs and replace this one with your content (use the “Normal” paragraph style).

Appendix B – Images

## Windows

### Welcome Window – Light Mode

Graphical user interface, text, application

Description automatically generated

### Welcome Window – Dark Mode

Graphical user interface

Description automatically generated

### 7.2.1 Main Window – Light Mode

Graphical user interface, application, Word

Description automatically generated

### 7.2.2 Main Window – Dark Mode

### A screenshot of a computer Description automatically generated with medium confidence



### Find/Replace Window – Light Mode

Graphical user interface, application

Description automatically generated

### Find/Replace Window – Dark Mode

A screenshot of a computer

Description automatically generated with medium confidence

### 7.4.1 Manual – Dark(Always Dark)

A screenshot of a computer

Description automatically generated with medium confidence

### 7.5.1 Compare Window – Light (Always Light)

Graphical user interface, application

Description automatically generated