Interim Report 2023

Text Editor

Interim Report

Table of Contents

[1. Introduction 3](#_Toc151018828)

[1.1 Plain text editors 3](#_Toc151018829)

[1.2 Rich Text Editors 4](#_Toc151018830)

[1.3 Objectives 5](#_Toc151018831)

[1.4 Language 5](#_Toc151018832)

[1.5 Code Editor Libraries 6](#_Toc151018833)

[2. Requirements Analysis 6](#_Toc151018834)

[2.1 User Interface 6](#_Toc151018835)

[2.2 Requirements 7](#_Toc151018836)

[2.2.1 Mandatory 7](#_Toc151018837)

[2.2.2 Desired 7](#_Toc151018838)

[3. Project Plan 7](#_Toc151018839)

[3.1 Phases and Tasks 7](#_Toc151018840)

[3.2 Completed Work 8](#_Toc151018841)

[4. Log 8](#_Toc151018842)

[4.1 Weekly Log 8](#_Toc151018843)

[4.2 Supervisor Log 9](#_Toc151018844)

[Bibliography 10](#_Toc151018845)

# Introduction

A text editor is a software application which allows a user to create and edit text files. Plain text editors are used by developers to write programs, and include features such as syntax highlighting, code completion, support for debugging and embedded git. Rich text editors are used to create documents, and allow users to change font colour and size, as well as add images and pages. This report will give an overview of the goals and objectives of this project, including the problems that need solving and research required.

## Plain text editors

**Vim**

A screenshot of a computer

Description automatically generatedOne example of a text editor is vim [1]. Vim is a command line text editor, which is included on most Unix based systems

Figure : Vim

Vim is an editor that will take longer to get used to than other text editors, due to its lack of a mouse-driven interface, modal nature and the need to learn specific commands and shortcuts. However, many users find that the initial steep learning curve of vim pays off in increased productivity.

**Visual Studio Code**

On the other end of the spectrum for plain text editors is Visual Studio Code (VSCode) [2]. VSCode is a very popular text editor, with a clean and intuitive user interface.

A screenshot of a computer

Description automatically generated

Figure : Visual Studio Code

It includes a sidebar for file navigation, an integrated terminal and a customizable layout. This provides a powerful and modern experience for a developer. VSCode also includes a rich set of features, including a vast library of extensions, git integration, a debugger, Live Share, and support for a wide range of languages.

## Rich Text Editors

Rich text editors, also known as word processors, are a different type of text editor. These editors allow you to edit text with a variety of formatting options beyond plain text. They are used for many purposes, including documentation, blogging, presentations and note taking.

A screenshot of a computer

Description automatically generated

Figure : TextEdit

**TextEdit**

One example of a rich text editor is TextEdit [3], which is a built-in rich text editor for macOS. It is a basic word processor, however, includes many features to help you format text. It also includes a plain text mode, which is suitable for coding and basic note taking. The interface is minimalistic and user-friendly, including a toolbar with formatting options.

**Microsoft Word**

Another word processor is Microsoft Word [4], which is one of the most popular and feature-rich word processors. It provides a comprehensive set of tools for creating, editing and formatting documents.

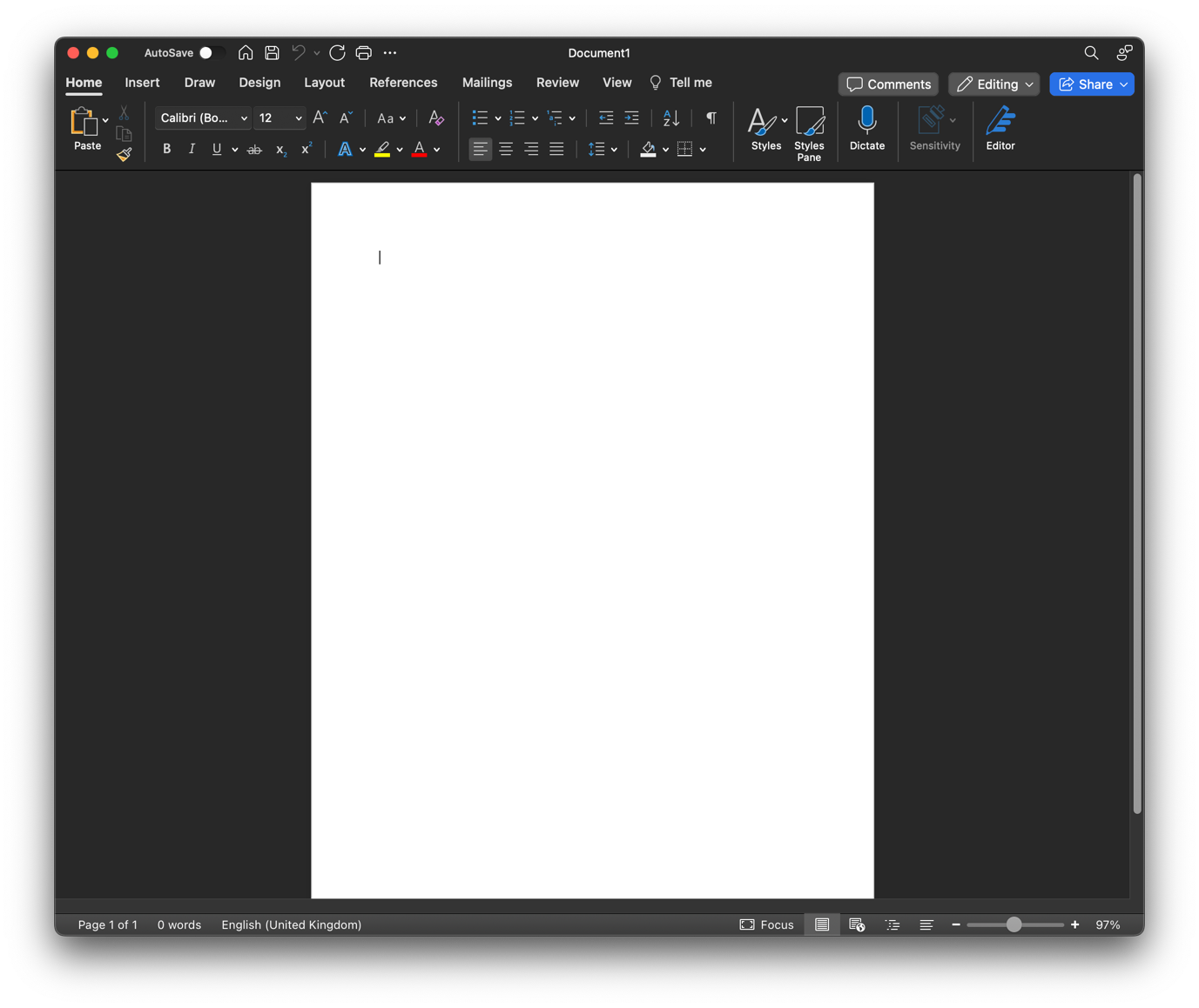
The user interface is user-friendly, and includes a toolbar at the top, providing a vast range of formatting and editing options.

Figure : Microsoft Word

## Objectives

The aim of this project is to create a plain text editor for developers to use. The ideal outcome of this project is to have a text editor which has a simple, user-friendly design, with as many features as possible. Although the text editor will be catered for programmers, it is not exclusive to them, and will be a great tool for all kinds of text editing.

## Language

The Electron [5] framework will be used to build this text editor, which will be used to create a desktop app through HTML, CSS and JavaScript. There are multiple reasons for using Electron to create a text editor. Firstly, Electron allows you to build applications that can run on different operating systems, including macOS, Windows and Linux. Secondly, it has a large ecosystem of libraries, plugins and tools available, which will help with including certain features in the text editor. As well as this, Visual Studio Code was built using Electron, which shows it can be used to create a feature-rich text editor with extensive features, and a highly customizable UI.

A potential downside to using Electron, is that the memory consumption can be high compared to native applications.

**Some key components of Electron:**

* Electron utilizes Chromium, which provides a fast and reliable platform for rendering web content in desktop applications.
* Node.js provides the backend functionality. This means developers can use the Node.js modules and npm packages to access the filesystem and manage processes.
* Electron consists of two main processes: the main process and the renderer process. The main process deals with the lifecycle of the application and interacts with the operating system. The renderer process controls the window in use and is responsible for rendering the web content and user interaction.

**Some key features of Electron:**

* Cross-platform compatibility
* Native desktop integration
* DevTools: Electron integrates with the Chrome Developer Tools, which provides familiar debugging tools.

## Code Editor Libraries

A code editor library would significantly simplify the process of creating a text editor in Electron.

Code editor libraries often come with a rich set of features that are included in most code editors. This includes syntax highlighting, code folding, line numbers, bracket matching and more. This would mean these features would not have to be coded from scratch, simplifying the process and allowing for further improvements. Customizable themes is also something that often comes with a code editor library, which would allow a user to personalise their experience.

Some examples of code editor libraries which can be used in Electron are:

* CodeMirror
* Ace
* Monaco Editor – the editor used in Visual Studio Code

# Requirements Analysis

## User Interface

One of the first requirements for the text editor is the layout. The layout will consist of a ribbon bar at the top for tabs, a navigation panel on the left for file operations, and the text input area on the rest of the screen.

## Requirements

The intended users for this text editor are programmers who require a platform which will include many features and a clean and easy to use user interface.

### Mandatory

1. Text editor runs on Windows and MacOS
2. Editing capabilities:
   1. Inserting and deleting text
   2. Copying and pasting text
   3. Undo/redo functionality
3. File operations
   1. Creating a new file
   2. Saving a file
   3. Opening an existing file
4. Multiple tabs/windows
5. Syntax highlighting
6. Plugin support

### Desired

1. Search and replace
2. Keyboard shortcuts
3. Customisation, for example theme choices
4. Auto-save

# Project Plan

The plan is critical for a successful project. This section will outline the tasks required to complete the project and a timeline, providing a roadmap for the development of the project.

## Phases and Tasks

1. Proposal
2. Planning and designing.
   1. Designing the text editor’s architecture
   2. Plan implementation of key features
   3. Explore and research Electron capabilities
3. Coding
4. Interim Report
5. Testing
6. Refinement and optimization.
   1. Refine the user interface to improve user experience
   2. Optimize performance and address potential bottlenecks
   3. Expand functionality
7. Final Report

Figure : Gantt Chart

Timetable to show when I will work on the project during the week:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Mon | Tue | Wed | Thu | Fri |
| 9:00 |  | Lecture |  |  |  |
| 10:00 |  | Lecture |  | Lecture |  |
| 11:00 | Lecture | Project |  |  |  |
| 12:00 | Lecture | Project | Project |  | Project |
| 13:00 | Project | Lecture | Project | Lab |  |
| 14:00 |  |  | Project | Lab |  |
| 15:00 | Seminar |  | Project |  |  |
| 16:00 |  |  |  |  |  |
| 17:00 |  | Lab |  |  |  |
| 18:00 |  |  |  |  |  |
| 19:00 |  |  |  |  |  |

This table shows the hours I have set aside to work specifically on the project. This will give me 8 hours per week. This timetable can change depending on the week, however the structure should be consistent throughout the year.

## Completed Work

I have completed most research fundamental to the text editor, however research will still be required for future stages of implementation. Research completed includes:

* Text editors. What requirements need to be met in order to build a good text editor? How are they designed?
* Electron. Understanding how it can help me, what it offers in terms of building a text editor, and how it works.
* Code editor libraries. Understanding how they can help me create more features and improve productivity.
* GitHub. Understanding how to use GitHub for version control.

I have also completed some work on the code, particularly the layout of the editor. Figure 6 shows the progress made on creating the layout. There is a sidebar on the left, which I will use for file navigation, and any additional features that require a sidebar. The ribbon along the top shows where the tabs will be shown, so if a user has multiple files open at the same time, they will be able to select which one to work on. The large white area is where the user will input text.

A screen shot of a computer

Description automatically generated

Figure : Layout

# Log

### Weekly Log

**Term 1 Week 1**

Before starting this project, I already had experience with using HTML, CSS and JavaScript, however I hadn’t used them in Electron before. After choosing the project and deciding to use Electron to code it, I started researching to understand the framework and how I can use it.

After researching, I started writing my project proposal.

**Term 1 Week 2**

* Finished project proposal
* Created a plan for tasks, drawing out when to start certain tasks and which tasks need to be completed

**Term 1 Week 3**

* Started design sketches to understand what I wanted the text editor to look like.
* Set up file structure and GitHub [6] repository

**Term 1 Week 4**

* Began coding the basic structure of the text editor
* Continued to research Electron

**Term 1 Week 5**

* Created the layout of the text editor content
* Research on code editor libraries such as CodeMirror and Ace.

**Term 1 Week 5**

* Continued to add code to text editor
* Research into file management
* Started interim report

**Term 1 Week 6**

* Continued to add code to text editor
* Improved code

**Term 1 Week 7**

* Completed interim report

## Supervisor Log

**22/10/23**

* Discussed initial project proposal, interim report and general project.
* Clarified expectations.

**11/11/23**

* Email to let supervisor know I had no questions.

# Bibliography

|  |  |
| --- | --- |
| [1] | Vim, “Vim Documentation,” Vim, 2023. [Online]. Available: https://www.vim.org/docs.php. [Accessed 13 11 2023]. |
| [2] | Microsoft, “Visual Studio Code,” Microsoft, 2023. [Online]. Available: https://code.visualstudio.com/. [Accessed 13 11 2023]. |
| [3] | Apple, “TextEdit User Guide,” Apple, 2023. [Online]. Available: https://support.apple.com/en-gb/guide/textedit/welcome/mac#:~:text=With%20TextEdit%2C%20you%20can%20open,open%2C%20edit%20and%20convert%20documents. |
| [4] | Microsoft, “Microsoft Word,” Microsoft, 2023. [Online]. Available: https://www.microsoft.com/en-gb/microsoft-365/word. |
| [5] | OpenJS, “Electron Docs,” Electron, 2023. [Online]. Available: https://www.electronjs.org/docs/latest. |
| [6] | GitHub, “GitHub,” GitHub Inc, 2023. [Online]. Available: https://github.com/. |
| [7] | B. Pickle, “Text Editor,” TechTerms.com, 10 October 2022. [Online]. Available: https://techterms.com/definition/texteditor. [Accessed 13 11 2023]. |

**Project Proposal**

**Text Editor**

By William Wakeford

Supervisor: Ian Mackie

There are a vast number of text editors you can choose from; some are catered for programming and some for writing documents. Companies like Microsoft and Adobe have created popular text editors which include an abundance of features, for example syntax highlighting, auto-completion and plugin availability. The goal of this project is to create a versatile text editor which includes useful features, customization for user experience, and good functionality.

A screenshot of a computer

Description automatically generatedSome examples of text editors designed for development are Visual Studio Code, Sublime Text, Emacs and Vim.

This is a screenshot from Visual Studio Code, a source code editor made by Microsoft. It includes features such as syntax highlighting, support for debugging, code completion, code refactoring, snippets and embedded git. This is a very popular text editor used among programmers. It has a clean and simple design like most code editors, with a window for the folder structure (and other options), the text editor, and a terminal.

**Aims**

Build a text editor with a UI which is easy to use, modern and customizable, and includes:

* Basic editing features (copy, paste, undo, redo, search, replace)
* Ability to open, edit and save files
* Syntax highlighting
* Auto completion
* File management
* Support for plugins
* Catered towards programming

**Objectives**

1. Research
2. Design
3. Build
4. Test
5. Improve