Deliverable 1

Problem Statement and Software Requirements

Group 4 — A CLI based Editor - Bloc++

Harsh Shah

Nihal Kulkarni

Nirmal Patel

Shivayashwanth Chikkondra

Course Instructor: Cristina Ruiz Martin

October 6, 2021

1 Problem Statement

For any beginner, who is just starting to learn coding, the first thing they have to do is configure the environment for the language they intend to use. Although there are many GUI based IDE (for example VS code, Eclipse, Sublime, CLion, CodeBlocks, Netbeans, etc.) which works seamlessly for almost all the languages, it might be difficult and also time consuming to download, install and configure the environment (setting the path variable or pointing to the appropriate compiler). For a novice, it may take a lot of time and effort before they write their first "Hello World" program.

In this case CLI based tools are more convenient to use. Almost all the operating systems come with built-in CLI. We just need to install compiler (for Linux based OS gcc compilers are preinstalled) and then open up a editor like vim and start writing the code. After saving the code it is easy to run the code in the same window and see the output there as well. So overall these CLI based tools require minimal efforts to start working on a program.

Not only for beginners even advanced users need CLI based editors to edit programs, scripts and system files. Most of the advanced users prefer CLI over GUI for 4 main reasons,

- **Requires less resources**: The CLI based editors need very little computer resources compared to GUI based editors. So if we have a computer with less primary memory or want to do tasks which requires high CPU performance, CLI tools are the right option.
- **Powerful**: CLI based tool have more power compared to GUI tools as the developer of GUI editor may have restricted access to some directories or files, but with CLI we have full control over the system.
- **Repetitive task friendly**: With GUI we can complete our tasks easily, but if we have to perform same task again and again, changing windows might leave us overwhelmed. With CLI we can do this tasks using a few commands or a few lines of script. For example, we can search for a text in all the files under the current directory with one CLI command.
- **Precision**: We can use specific commands to target specific tasks with ease. As long as we use the correct command, the chances of making errors are less compared to GUI.

Therefore, CLI based editors are useful for all levels of programmers. In the next section the purpose of this software is described in detail.

2 Software Purpose

In this project, we propose to implement a CLI based editor, which will be suited for creating and editing program files. Through this editor, users will be able to write and edit program files with syntax highlighting, auto-indent, auto-closing brackets and search and highlight keywords in the file.

Secondly, this text editor allows to edit program files, highlights the current syntax/line, and gives indentation programming. Thereby, helping the user to convey the code in a structured manner. Furthermore, error spotting and debugging will be easy, creating and editing files will be simple. It will have 2 modes of operation (after opening the file), insert mode and command mode.

The purpose of this software is to build an editor which is easy to understand and requires minimal resources and time to start writing the codes. This software will help beginners to start writing their first code without putting in too many efforts. Moreover, this software will be a handy tool for the experienced professionals if they want to edit any system file which requires special permissions or just want to make changes to their existing code. Compiling and running the program/code is trouble-free. In addition, a less intricate single toolbar makes it more visually appealing. Also, absence of completion of keywords automatically gives a true sense of learning and helps in understanding the thought process of programming. Hence, results in improving the debugging skills, analyzing, and fixing the misses.

3 Software Requirements

Any software can be delivered on time when the demands are tangible and in correct priority order. Below we'll look at a few requirements for the text editor from a client's perspective. In the next section, there is a talk on releases where we'll be highlighting when the aforementioned demands would be sorted out.

1. Users should be able to create a new file by giving the file name and with extension. If the file already exists in the current directory, open that file and user should be able to edit the contents of the file.

2. User should have two modes one for editing the content of the file (insert mode) and command mode to save the file, search in the file and quit the keywords.
3. The text editor should have a color highlighting feature for all the semantics in order to differentiate regular text from code files.
4. Implement a feature of keyword search which highlights all occurrences of the keyword, as it's really necessary while troubleshooting or editing a file.
5. Keeping a note on the user's productivity, auto-completion of braces and brackets must be included.

4 Software Releases

The software will be released in two versions.

With the initial build, users can start using this software with some preliminary features of creating new files, editing existing ones and performing save/quit operations using software commands.

In the second and the final release, users will have a complete package with everything they need in order to successfully create an impact in their software productivity including color highlighting, keyword search and auto-completion of brackets and braces.