Introduction

While developing in terminals without a graphical interface. Although there are powerful text editors such as Vim and Nano, we often still find it hard to modify text files because key shortcuts are different. To make it easier for users, we created a npm text editor library, it supports all basic features of a text editor, and also allows user to customize key short cuts.

Overview of Features and Purpose

Basic functionalities

The editor supports all basic operations. It allows the user to type in characters, tabs, new lines and spaces. It also supports cursor movements. To find out more about this, you can take a look at the documentation.

Acceptance criteria

- Typed characters produce the corresponding character at the cursor position and advance the cursor position
- Enter adds a newline and moves the cursor to the start of the next line
- Left arrow key moves the cursor one character left, or the end of the previous line if at the beginning of a line
- Right arrow key moves the cursor one character right, or the start of the next line if at the end of a line
- Up arrow key
- Down arrow key

Open and Save

The editor can also open an existing file and make modifications to it. It can also save the current opening file as a new file.

Acceptance criteria:

- Passing the path to a file to the init method or as the first argument to the CLI opens the specified file
- The save shortcut writes out the contents of the editor’s buffer to the same file

Undo and Redo

The text editor supports Undo and Redo just as other text editors. The user can press CTRL_Z to undo and CTRL_Y to redo.

Copy and Paste

While in the terminal mode, the user can hold shift and use the cursor to select a range of text, after that, the user can press F2 to copy it, and then they can use CTRL_V to paste it into the terminal.

Acceptance criteria:

- Terminal-dependent paste shortcuts add text contents of the OS-dependent clipboard to the editor buffer at the current cursor position
- The highlight left shortcut extends the selected region one character to the left to the new cursor position
- The highlight right shortcut extends the selected region one character to the right to the new cursor position
- The copy shortcut overwrites the contents of the OS-dependent clipboard with the text in the currently highlighted region

Find Occurence

Our editor also supports Find. This functionality is particularly useful when the document is very long and you are only interested in finding some keywords. To do this, the user can press CTRL_F to activate the search and type in the key word.

Shortcuts

Shortcuts can be viewed by pressing CTRL+Q.
User can update the shortcut, for example, the user can enter the command mode and enter “update_shortcut UNDO”, then UNDO would be updated to the next non-character key you pressed
User can reset all shortcuts by entering the command “reset_shortcuts”
Note: CTRL + C (Terminate) can not be modified

Talked with our industry partner and decided to added three more features as our milestone 3

Repeating

The repeating functionality can be used when you want to repeat a set of input many times. For example, maybe the user wants to enter “print(“I love CSC302”)” 100 times. Instead of copying and pasting 100 times, the user can enter the command mode and enter “Repeat 100” “print(I love CSC302)”. This will have the same effect.

Recording

The recording functionality allows you to record some input and use them later. This is useful when a user needs to enter the same thing over and over again. For instance, maybe for

debugging purposes, someone needs to enter “assert(false)”, instead of having to type this every time, the user can use a shortcut to enter it.

Matching Parentheses

When moving the cursor on top of an open/close parentheses, the other matching one will be highlighted and underlined.

Features finished and group member’s contribution in Milestone 3

Repeating
Recording
Matching parentheses
Yujie finished Repeating and fixed bugs regarding cursor movement and shortcut.
David finished Matching parentheses
Tony finished Recording
Kevin finished automated testing

Acceptance and Validation Criteria

- For each functionality we aim to finish, we have corresponding documentation, in the documentation, we clearly state the functionality and directions on how to use them.
- Automated tests using the Jest framework cover most developed features

Features not delivered

There are only a few basic features we haven't implement

- replace (This feature is extremely difficult to implement due to the terminal-kit's text buffer design. After a in-team discussion, we decide to move on with other features)
- Show line numbers (This will drastically change the logic for finding the parentheses due to the change in screen buffer)