

# NEA - Productivity tool

Analysis	10
Introduction	10
Prospective users	11
Client background	11
Client interview	11
Existing products	12
Alfred	12
Advantages of product and how I can learn from it	13
How I can improve upon the product for my clients needs	13
Chrome bookmarks	14
Advantages of product and how I can learn from it	14
How I can improve upon the product	15
Rofi	15
Advantages of product and how I can learn from it	15
How I can improve upon the product	16
Keywords	16
<option-string>	16
Description	16
Representations	16
<identifier><action-delimiter><action-identifier><data-delimiter><data>	16
<identifier><action-delimiter><action-identifier>	16
<identifier-name><identifier-extension>	16
<identifier>	16
<name>	16
Description	16
Representation	17
<identifier>[<action>]	17
<data>	17
<action-delimiter>	17
Description	17
Representation	17
Types	17
<program-action-delimiter>	17
Description	17
Representation	17
<destination-action-delimiter>	17
Description	17
Representation	17
<menu-action-delimiter>	18
Description	18
Representation	18
<script-action-delimiter>	18
Description	18
Representation	18

<action-identifier>	18
Description	18
Types	18
<program-action-identifier>	18
<destination-action-identifier>	18
<menu-action-identifier>	18
<script-action-identifier>	18
<program-action-identifier>	18
Description	18
Types	18
<destination-action-identifier>	19
Description	19
Types	19
<menu-action-identifier>	19
Description	19
Types	19
<script-action-identifier>	20
Description	20
Examples	20
<action>	20
Description	20
Representation	21
<action-delimiter><action-identifier>	21
Types	21
<program-action>	21
<destination-action>	21
<menu-action>	21
<script-action>	21
<program-action>	21
Description	21
<destination-action>	21
Description	21
<menu-action>	21
Description	21
<script-action>	21
Description	21
An user coded script in the default script directory.	21
Description	21
<identifier>	21
Description	21
Representation	22
<identifier-name>[<identifier-extension-delimiter><identifier-extension>]	22
<identifier-extension-delimiter>	22
Description	22
Representation	22
<identifier-name>	22
Description	22
Examples	22

<identifier-extension>	22
Description	22
Examples	22
{location}	23
Description	23
<default-action>	23
Description	23
{scripts-set}	23
Description	23
{extension-to-action-dictionary}	23
Description	23
Example	23
<extension-linked-action>	24
Description	24
{input-text}	24
Description	24
{permission-start-bracket} and {permission-end-bracket}	24
Description	24
{mode-character}	24
Description	24
Syntax design	24
My initial thought process	24
Logic of program	25
Start of program	25
Parsing option string	25
Configuration directory	27
Tree view of example config directory and descriptions of each subdirectory/file	27
Example IDENTIFIER_EXTENSIONS file	28
Menu interface design	28
Layout	28
Title bar	29
Format	29
Example	29
Input box	29
Search option	29
Selected option	29
Status bar	29
Modes of menu	30
Edit mode	30
Insert mode	30
Search mode	30
Types of menu	30
Bookmark menu	30
Data menu	30
Directory menu	30
Menu system example	30
Menu 2	31
Menu 3	31

Menu 4	31
Objectives	32
General objectives	32
Extension objectives	34
Objectives about program specifics	34
Alternative solutions and justification of chosen solution	36
Alternative solutions	36
Chosen solution	36
Acceptable limitations	36
<b>Documented design</b>	<b>37</b>
Class relationships	37
MenuTUI Class	37
MVC design pattern	37
MenuData	39
MenuController	40
MenuView	41
History stack	42
OptionList class	42
Search algorithm	43
Tokenized matching algorithm	43
Pseudocode	44
FileOptionList	45
InputOnlyOptionList	45
Log class	45
Singleton design pattern	45
Fixed length queue	45
Implementation	46
Builder design pattern	46
History and log data structure	46
Flushing history and data	46
Pseudocode	46
Parser Class	47
Initialization	48
Data flow diagram of execution	48
Finding position of action using recursion	48
String to function dictionaries	49
Pseudocode	51
<b>Technical solution</b>	<b>51</b>
config_directory.hpp	51
config_directory.cpp	51
exit_code.hpp	53
log.hpp	55
log.cpp	55
menu_tui.hpp	60
menu_tui.cpp	63
menu_controller.hpp	64
read_only_menu_controller.hpp	66
editable_menu_controller.hpp	67
menu_controller.cpp	69

editable_menu_controller.cpp	69
read_only_menu_controller.cpp	70
component_controller.hpp	72
cursor_position_controller.hpp	76
cursor_position_controller.cpp	76
input_controller.hpp	77
input_controller.cpp	77
option_list_controller.hpp	79
option_list_controller.cpp	82
selected_option_position_controller.hpp	83
selected_option_position_controller.cpp	85
title_controller.hpp	86
title_controller.cpp	87
menu_view.hpp	87
menu_view.cpp	87
menu_data.hpp	89
option_list.hpp	93
data_option_list.hpp	95
bookmark_option_list.hpp	98
bookmark_option_list.cpp	99
editable_directory_option_list.hpp	101
editable_directory_option_list.cpp	104
file_option_list.hpp	105
input_only_option_list.hpp	107
read_directory_option_list.hpp	108
read_directory_option_list.cpp	109
read_only_data_option_list.hpp	109
scripts_directory_option_list.hpp	110
scripts_directory_option_list.cpp	110
option_list.cpp	111
data_option_list.cpp	112
bookmark_option_list.cpp	115
read_directory_option_list.cpp	116
editable_directory_option_list.cpp	120
scripts_directory_option_list.cpp	120
menu_data.cpp	122
parameter_processor.hpp	124
parameter_processor.cpp	124
parser.hpp	125
parser.cpp	125
retriever.hpp	129
file_retriever.cpp	136
file_retriever.hpp	137
directory_retriever.hpp	138
directory_retriever.cpp	138
<b>Testing</b>	<b>138</b>
Config directory	138
IDENTIFIER_EXTENSIONS file	139
SCRIPTS directory	139

Table describing what each script does	140
Evidence and contents of scripts	140
Testing environment	140
Testing core objectives	142
Objective: 1.a	142
Test	142
Expected result	142
Actual result	142
Objective: 1.b	142
Test 1	142
Expected result	142
Actual result	142
Objective: 1.c	142
Test 1	142
Expected result	142
Actual result	142
Objective: 2	143
Test 1	143
Expected result	143
Actual result	143
Objective: 3.a.i + 4.a.i + 5.a.i	143
Test 1	143
Expected result	143
Actual result	144
Test 2	144
Expected result	144
Actual result	144
Test 3	144
Expected result	145
Actual result	145
Objective: 3.a.ii + 4.a.ii + 5.a.ii	145
Test 1	145
Expected result	145
Actual result	145
Test 2	145
Expected result	145
Actual result	145
Test 3	146
Expected result	146
Actual result	146
Objective: 3.a.iii	146
Test	146
Expected result	146
Actual result	146
Objective: 3.a.iv	146
Test	146
Expected result	147
Actual result	147
Objective: 3.a.v	147

Test	147
Expected result	147
Actual result	147
Objective: 3.a.vi	147
Test	147
Expected result	147
Actual result	147
Objective: 3.a.vii	147
Test 1	147
Expected result	147
Actual result	147
Test 2	148
Expected result	148
Actual result	148
Objective: 3.a.viii	148
Test	148
Expected result	148
Actual result	148
Objective: 3.a.viii	148
Test 1	148
Expected result	148
Actual result	148
Test 2	148
Expected result	148
Actual result	148
Objective: 3.a.x	149
Test 1	149
Expected result	149
Actual result	149
Test 2	149
Expected result	149
Actual result	149
Objective: 3.a.xii	149
Test	149
Expected result	149
Actual result	149
Objective: 3.a.xiii	149
Test 1	149
Expected result	149
Actual result	149
Objective: 3.a.xiv	149
Test 1	149
Expected result	150
Actual result	150
Objective: 3.a.xv	150
Test 1	150
Expected result	150
Actual result	150
Test 2	150

Expected result	150
Actual result	150
Objective: 3.a.xvi and 2.a.xvii	150
Test 1	150
Expected result	150
Actual result	150
Objective: 3.b.i	150
Test 1	150
Expected result	150
Actual result	151
Objective: 4.a.iii	151
Test 1	151
Expected result	151
Actual result	151
Objective: 4.a.iv	151
Test 1	151
Expected result	151
Actual result	151
Objective: 4.a.v	151
Test 1	151
Expected result	151
Actual result	151
Objective: 4.b.i	151
Test 1	151
Expected result	151
Actual result	151
Objective: 4.b.ii	152
Test 1	152
Expected result	152
Actual result	152
Objective: 4.b.iii	152
Test 1	152
Expected result	152
Actual result	152
Objective: 5.a.iii	152
Test 1	152
Expected result	152
Actual result	152
Objective: 5.a.iv	152
Test 1	152
Expected result	152
Actual result	152
Objective: 5.b.i	153
Test 1	153
Expected result	153
Actual result	153
Objective: 6.a.i - iii	153
Test 1	153
Expected result	153



Actual result	153
Objective: 6.b.i - iii	153
Test 1	153
Expected result	153
Actual result	153
Objective: 6.c	153
Test 1	153
Expected result	153
Actual result	153
Objective: 7.a + 7.b + 7.c + 7.d	153
Test	153
Objective: 8.a	154
Test 1	154
Expected result	154
Actual result	154
Test 2	154
Expected result	154
Actual result	154
Objective: 8.b	155
Test 1	155
Expected result	155
Actual result	155
Testing extension objectives	155
Objective: 9.a	155
Test 1	155
Expected result	155
Actual result	155
Objective: 9.b	155
Test 1	155
Expected result	155
Actual result	155
Objective: 10	155
Objective: 11	155
Test 1	155
Expected result	156
Actual result	156
Why test failed	156
Test 2	156
Expected result	156
Actual result	156
Objective: 12	156
Objective: 13.a	156
Test 1	156
Expected result	156
Actual result	156
Objective: 13.b	156
Test 2	156
Expected result	156
Actual result	156

Objective: 13.c	157
Test 1	157
Expected result	157
Actual result	157
Objective: 13.d	157
Test 1	157
Expected result	157
Actual result	157
Objective: 14 and 15	157
Test	157
Objective: 16.a	157
Test 1	157
Expected result	158
Actual result	158
Test 2	158
Expected result	158
Actual result	158
Objective: 16.b	158
Test 1	158
Expected result	158
Actual result	158
Objective: 16.c	158
Test 1	158
Expected result	158
Actual result	158
Objective: 17.a	158
Objective: 17.b	159
Objective: 17.c	159
Objective: 17.d	159
Objective: 17.e	159
Objective: 17.f	159
Testing evidence video	159
<b>Evaluation</b>	<b>159</b>
Client feedback	159
Follow up questions	159
How effective was the system I designed in solving my problem	159
Potential improvements	160

## Analysis

### Introduction

Desktop users use a computer for a variety of tasks, e.g. reading documents, watching videos and reading the news. Users may want to directly run tasks they perform regularly, bypassing steps such as opening the browser to open a website they often view or opening a file manager to open a document they are reading.

For tasks they perform less regularly, it is often hard to remember information required to perform the task, such as the location of the file they want to open or the name of the website they want to read. If they are able to easily save such tasks as bookmarks with meaningful names as well as group related tasks together, it could save them a lot of time.

This project aims to create a menu framework where the user can easily save and group together different computer tasks as searchable bookmarks with user-given names, ensuring they are memorable.

## Prospective users

The main group this project will be designed for are GCSE students. As a result of the range of different formats that learning resources are distributed by teachers - PDFs, Onenote pages, random links, word documents, etc - It is difficult to aggregate them and organise them together.

Also, due to the quantity and the variety of the resources, it is difficult to find those resources when one is revising for exams when one has not used those resources for a while.

My project will allow the user to save resources and notes of different topics together, helping them be more immersed in their revision, improving their focus and productivity.

## Client background

John Thompson uses computers on a daily basis to perform a multitude of tasks. He uses an Ubuntu laptop.

He's currently a Year 11 student whose school transitioned last year to compulsory use of computers during lessons to take notes. Teachers use a variety of software to distribute resources, such as OneNote, Teams and Sharepoint and a variety of software to set work, such as DrFrostMaths and Isaac Physics.

He has topic assessments roughly every month for each of his 9 subjects, and end of year exams towards the end of each school year. To revise for these tests, he uses Word handouts, PDF questions sheets and specifications, online topic notes, his own OneNote notes and Youtube teaching videos.

He would like a way to collate materials of each topic together, but hasn't found a suitable method.

Besides school, he uses the computer to play online chess, watch Youtube videos and to listen to music.

## Client interview

**How do you currently take notes?**

I usually use OneNote as that is where most teachers put the bulk of the materials. However, I don't like its user interface and some teachers, like our Biology teacher, puts everything on Teams.

### Describe your revision process

Well it is different for different subjects. Well ... for all the subjects I often look through our topic checklist, which is located on our student portal and find my weak areas. For subjects where I have to remember a lot of facts, like Geography, I use flashcards I make on Quizlet to revise. For science subjects, I do a mix of reading notes and doing GCSE questions I find, on like PhysicsAndMathsTutor, to apply my knowledge.

### How do you organise your revision times to ensure you get sufficient study times of each topic?

I use a method called spaced repetition. It is an effective way of revising in which I come back to a topic in spaced amounts of time, making sure I remember stuff in the long term. I have a study calendar where I plan I when I will do each topic, which I have a systematically choose topics based on when I learnt them and how confident I am of each one

### Have you ever been put off from revision due to not knowing where resources were?

Yeah, I think I have. The effort needed to regather resources to effectively revise has definitely caused me to procrastinate. I guess I should organise my notes better.

### Describe the differences between using the computer for school and using books, and specify advantages/disadvantages of either

Since we started using computers, it has been very disorganised, with different resources in different locations. The teachers often don't initially know where the resources are. I assume this is because it is the first year though, and they are still getting used to it. I think I've had an overall better experience of using books, since everything is in one place when you are revising. However, it is handy to take notes on a computer, being able to copy diagrams and text.

### Do you ever find it tedious to open the browser or file manager to open commonly opened links or files?

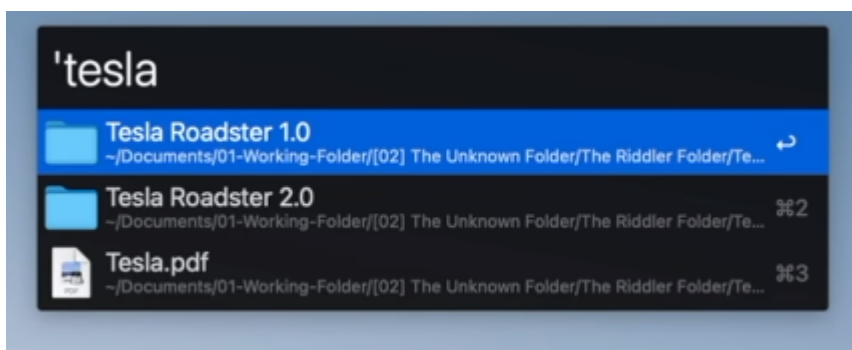
Yeah definitely, I feel that there should be a feature on app launchers to save these to be directly openable.

## Existing products

I found several application software programs that serve similar functions to what my project aims to do and analysed them.

### Alfred

Alfred is a productivity application for macOS that allows users to launch apps, search the web, and perform system commands with just a few keystrokes. Alfred provides a quick and efficient way to access various features of your computer, saving users time and effort using a searchable menu and customisable key shortcuts.



Different keyboard shortcuts correspond to different menus.

- CMD-SPACE will open the default results menu with all your applications and system commands, and you are able to extend this with a config menu.
- CMD-SPACE-SPACE will open a menu with all your files.
- CMD-SPACE-Keyword can connect search results in browser, for example, yt will connect the search results to youtube's autofill

Note that this is just a very brief overview of the high level functionality of Alfred, which contains many more features I haven't touched on.

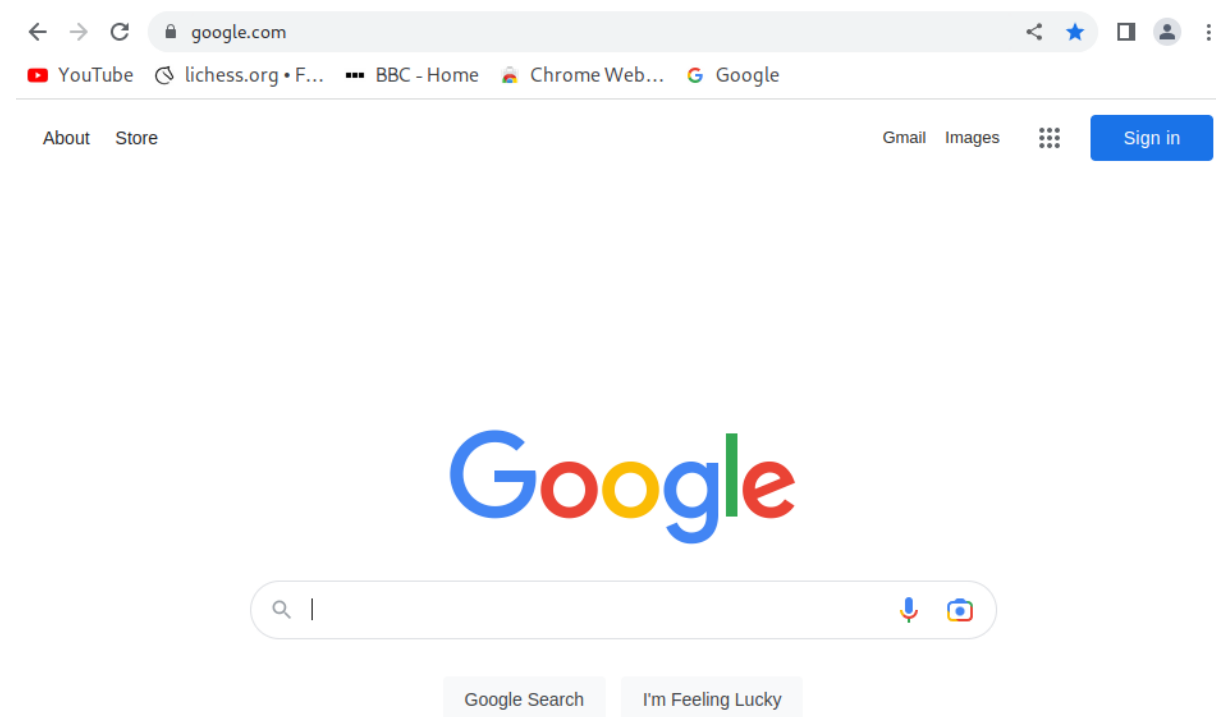
### Advantages of product and how I can learn from it

It is highly configurable, allowing users to add different sets of bookmarks separately. You can also allow direct access to different features such as google search, which is a feature my client says is what he uses a computer most commonly for. Much of this convenience is due to its use of keywords, so I can try and design a system that does something similar.

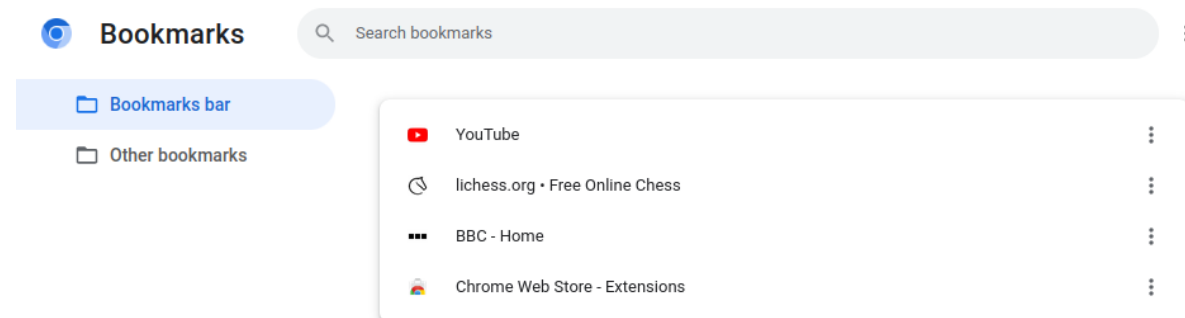
### How I can improve upon the product for my clients needs

Due to its many features, it can be overwhelming to learn and use to maximum effect. My project will try to allow users to directly create features using their own logic, perhaps by allowing them to write their own scripts to generate actions, so that the user can add exactly what they want and know how to get it.

## Chrome bookmarks



Chrome bookmarks is a feature integrated into the Chrome browser. It allows the user to save and have easy access to commonly accessed links. To create a bookmark, you navigate to the link you want to save, and press the star icon on the top right of the address bar. It then gives you the option to choose the location of where to save it, by default being the top level folder Bookmarks bar which represents the folder in which the shortcut is being directly displayed.



Another feature of Chrome bookmarks, is that when typing @bookmarks into your address bar or pressing CTRL-SHIFT-O, the bookmarks become searchable in the address bar and in a separate bookmarks tab respectively.

### Advantages of product and how I can learn from it

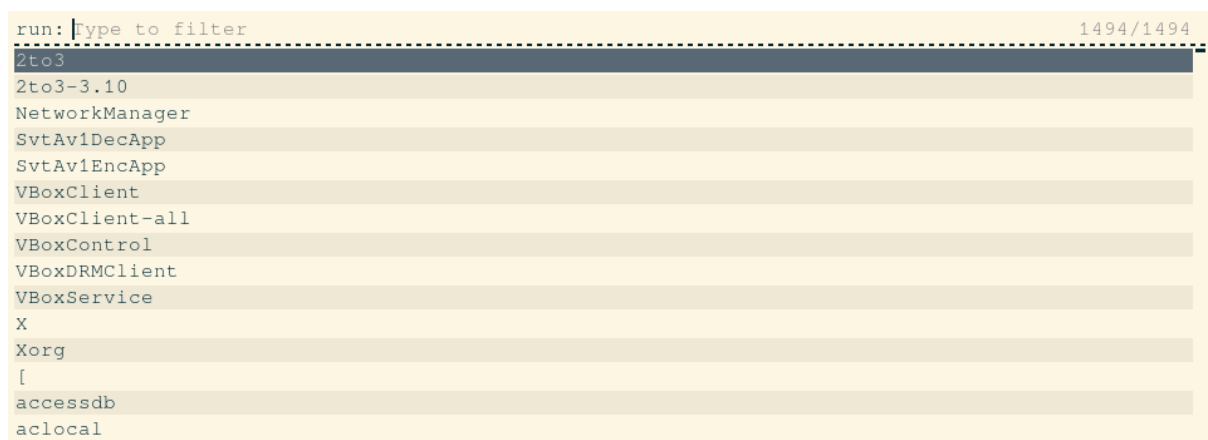
The ability to search allows for easy access of different bookmarks despite the user perhaps forgetting the specific names, saving them the time to look through bookmarks one by one. I want to integrate that into my bookmarks system as the main way of accessing bookmarks.

I also want to incorporate the ability to group bookmarks together. This ensures that the system remains organised and manageable so that the user can remain efficient despite having a lot of bookmarks.

### How I can improve upon the product

Its integration directly into the browser allows a simple and intuitive interface to create and access bookmarks. However, a bookmark system integrated directly into the browser also has its limitations, namely that it only saves links and can only open them in this browser. Also, as it is not the main focus of the program, it must accommodate for other features, meaning only a few bookmarks can be saved in the view of the user and the search function is not directly there after opening the browser.

### Rofi



Rofi is a lightweight and efficient application launcher and window switcher for Unix-like systems. It provides a quick and convenient way to access and switch between open windows, launch applications, and perform other tasks with a few keystrokes. Rofi is highly customizable, with a wide range of configuration options that allow users to tailor it to their specific needs and preferences. It is also extensible, with support for plugins that can add additional features and functionality.

Similar to Alfred, there are multiple types of menus that can be opened, including:

- run - opens executables in \$PATH
- drun - opens .desktop files
- filebrowser - opens a file
- window - allows user to switch windows

The menus must be binded by the key using a separate program by the user, meaning it is slightly less user friendly to set up, but on the other hand allows for greater configurability.

### Advantages of product and how I can learn from it

Its appearance and functionality is highly customizable using a config file, which would allow users to easily change it to match their needs.

Also, the fact that it allows different types of menus to be opened where each option represents a different thing is interesting and maybe something I can take inspiration on.

## How I can improve upon the product

Listing out all of the executable files is convenient, however, there is a lot of clutter that are invalid. My program could allow users to add and remove their own options.

## Keywords

The following keywords will be referred to throughout the course of the project.  
Text surrounded by "<>" represents that it refers to some text in an option string.  
Text surrounded by "{}" represents a general keyword.

### <option-string>

#### Description

Whole string that is executed. Each <option-string> must always have a corresponding <data> and <action>.

#### Representations

<identifier><action-delimiter><action-identifier><data-delimiter><data>

**Condition:** <data-delimiter> found, <action-delimiter> found, and <action> is valid

<data>: <data>

<action>: <action-delimiter><action-identifier>

<identifier><action-delimiter><action-identifier>

**Condition:** <data-delimiter> not found, <action-delimiter> found, and <action> is valid

<data>: <identifier>

<action>: <action-delimiter><action-identifier>

<identifier-name><identifier-extension>

**Condition:** <data-delimiter> found, <action-delimiter> found, and <action> is valid

<data>: <data>

<action>: <extension-linked-action>

<identifier>

**Condition:** <action> is invalid or not found or <option-string> is an option in a menu which forces a <default-action>

<data>: <identifier>

<action>: <default-action>



## <name>

### Description

What is shown to the user to identify an entry. Each option in a menu must have a unique <name> in each entry. <name> depends on the type of menu that has been opened.

### Representation

<identifier>[<action>]

**Condition:** bookmark-menu> or <list-menu> has been opened

<data>

**Condition:** menu that is not bookmark-menu has been opened.

## <action-delimiter>

### Description

Specifies where the start of an <action> and what type of <action-identifier> follows it.

### Representation

<action-delimiter><action-identifier>

### Types

<menu-action-delimiter>

<program-action-delimiter>

<destination-action-delimiter>

<script-action-delimiter>

## <program-action-delimiter>

### Description

A type of <action-delimiter>. Specifies where the start of a <program-action-identifier>

### Representation

~

## <destination-action-delimiter>

### Description

A type of <action-delimiter>. Specifies where the start of a <destination-action-identifier>

Representation

@

<menu-action-delimiter>

Description

A type of <action-delimiter>. Specifies where the start of a <menu-action-identifier>

Representation

:

<script-action-delimiter>

Description

A type of <action-delimiter>. Specifies where the start of a <script-action-identifier>

Representation

|

<action-identifier>

Description

Combined with <action-delimiter>, it defines an <action>.

Types

<program-action-identifier>

<destination-action-identifier>

<menu-action-identifier>

<script-action-identifier>

<program-action-identifier>

Description

A type of <action-identifier>. Specifies the type of the <program-action> is to be run.

Types

<program-action-identifier>	What it does
-----------------------------	--------------

option_string	Evaluates data as an option string, used to simplify program logic
echo	Echoes data, used for debugging
nothing	Does nothing, if no <extension-linked-action> found, this will print out that nothing happened as no default specified.

## <destination-action-identifier>

### Description

A type of <action-identifier>. Specifies the type of <destination-action> that is to be run.

### Types

<destination-action-identifier>	What it does
bmk	Opens a file where each line is an option and everything after the first '>' is hidden.
lst	Opens a file where the string on every line is the <data> and the <name>
rlst	Same as "lst" but file is read only.
dir	Opens a directory where every filename is an option. Every option's <data> will be its absolute path and every option's <name> is its filename.
rdir	Same as "dir" but directory is read only.
sdir	Same as "dir" but files created are all executable

## <menu-action-identifier>

### Description

A type of <action-identifier>. Specifies the type of <menu-action> that is to be run.

### Types

<menu-action-identifier>	What it does
flashcard	Toggles between <data> and <name> if option selected. If no option selected, adds to <menu>
setting	Toggles between <data> and <name> if option selected. If no option selected, adds to <menu>

todo	Removes currently selected option if there is an option selected, otherwise adds <input-text> to <menu>
scratchpad	Selects data and enters edit mode if option selected otherwise adds <input-text> to <menu>

## <script-action-identifier>

### Description

A type of <action-identifier>. Specifies the name of the script in the default location in the configuration directory that is to be run.

### Examples

<script-action-identifier>	What it does
link	Data should be a link Opens the link in a web browser
search	Searches the web with data
app	Data should be an application Opens the application

## <action>

### Description

Specifies the exact operation that will be executed on data

## Representation

<action-delimiter><action-identifier>

## Types

<program-action>

<destination-action>

<menu-action>

<script-action>

### <program-action>

#### Description

A hard coded operation used for logic or debugging purposes, not designed to be used explicitly by the user for bookmarks.

### <destination-action>

#### Description

An hard coded operation that is run by the program to open a type of menu specified by its <action-identifier>.

### <menu-action>

#### Description

An hard coded operation that can only be run in a menu. It operates on the currently opened menu, and has a different implementation for if there is a selected option or if there isn't.

### <script-action>

#### Description

An user coded script in the default script directory.

#### Description

An <action>. Each menu has one, by default it is the <program-action> - "~option\_string." This <action> will be executed on the data of the chosen <option-string>.

## <identifier>

### Description

Memorable identifier given by user. Its value does not affect how it executes unless no data is specified, in which it replaces the name.

### Representation

<identifier-name>[<identifier-extension-delimiter><identifier-extension>]

## <identifier-extension-delimiter>

### Description

Specifies start of extension of file or link.

### Representation

.

## <identifier-name>

### Description

Beginning of a file name or link, the string before the final <identifier-extension-delimiter>.

### Examples

<option-string>	<identifier-name>
https://google.com	https://google
HOWTO.txt	HOWTO

## <identifier-extension>

### Description

The extension of a file name or link, the string after the final <identifier-extension-delimiter>. Empty if no <identifier-extension-delimiter>. Used as a key to {extension-to-action-dictionary} to get an action if one is not explicitly specified.

### Examples

<option-string>	<identifier-extension>
-----------------	------------------------

google.com	com
HOWTO.txt	txt
hello world	
bbc.co.uk	uk

## {location}

### Description

Fully defined `<action>`. Each time a destination action opens a menu a {location} is created. It represents the absolute path of the `<data>` where a menu was loaded from if the data was a path, otherwise it is just the `<data>`. This is so that it will refer to the same value even if the current directory is changed when it is stored in history.

## <default-action>

### Description

An `<action>`. When no `<action>` is explicitly specified, and there is no `<linked-extension-action>`, the `<default-action>` will be executed. By default, the `<default-action>` is the `<program-action>` - "`~nothing`", but if there is an `<identifier-extension>` as a key equal to an empty string in the {extension-to-action-dictionary}, that will be set to the `<default-action>`

## {scripts-set}

### Description

A set of all scripts loaded from SCRIPTS directory in configuration directory

## {extension-to-action-dictionary}

### Description

A dictionary loaded from a config file. This maps `<identifier-extension>`s to their respective `<action>`s.

### Example

<code>&lt;identifier-extension&gt;</code>	<code>&lt;action&gt;</code>
com	link
txt	text
	search

Note that the bottom row has an empty <identifier-extension>. This means that the <action> it is mapped to is the <default-action>.

## <extension-linked-action>

### Description

The action corresponding to the <identifier-extension> of the option string in the {extension-to-action-dictionary}

## {input-text}

### Description

Current text in the input box of the menu.

## {permission-start-bracket} and {permission-end-bracket}

### Description

Identifies permissions of current menu, whether it is read only or editable. If current menu is read only, brackets surrounding are "[ ]", else if menu is editable, they are "{}".  
{mode-character}:

## {mode-character}

### Description

Identifies what mode the current menu is in.

Character	Mode
'E'	Edit mode
'I'	Insert mode
'S'	Search mode

## Syntax design

### My initial thought process

After researching those products, I have discovered the convenience of keywords. I would like to extend it to a simple and readable syntax in which bookmarks can be saved. The syntax will allow the client to construct different actions that are all related to a task - such as opening links, apps, files, and group them together.

It would be difficult for me to hard code it into my program. Therefore, to facilitate all these different types of actions, I have opted for a more user extensible approach of using scripts.



The scripts names, located inside a configuration directory, will be loaded as the program starts and should take in one argument. The user can make scripts such as "link" which will open a link, or "app" which will open an app.

My initial idea was just to save bookmarks as follows: `{argument of script}|{script name}` with the '|' character being the final one found so that there are no conflicts due to the argument containing a '|' character. For example `chrome|app` will call the app script on chrome, so will open the chrome browser. Also, I thought to create a dictionary which the user can configure to link each extension with a script, so that if '|' is not found, it will look at the map to assign an appropriate script, and if that is not found, a default script specified by the user will be run. For example, if the dictionary has an entry `[com: link]`, `youtube.com` will be evaluated as `youtube.com|link`

However, my client pointed out that that may lead to long, complicated and incoherent names, which would defeat the point of this bookmark system, for example for opening Youtube links like - "<https://www.youtube.com/watch?v=m3dRlTirgLU>", writing "<https://www.youtube.com/watch?v=m3dRlTirgLU>|link" is pretty meaningless.

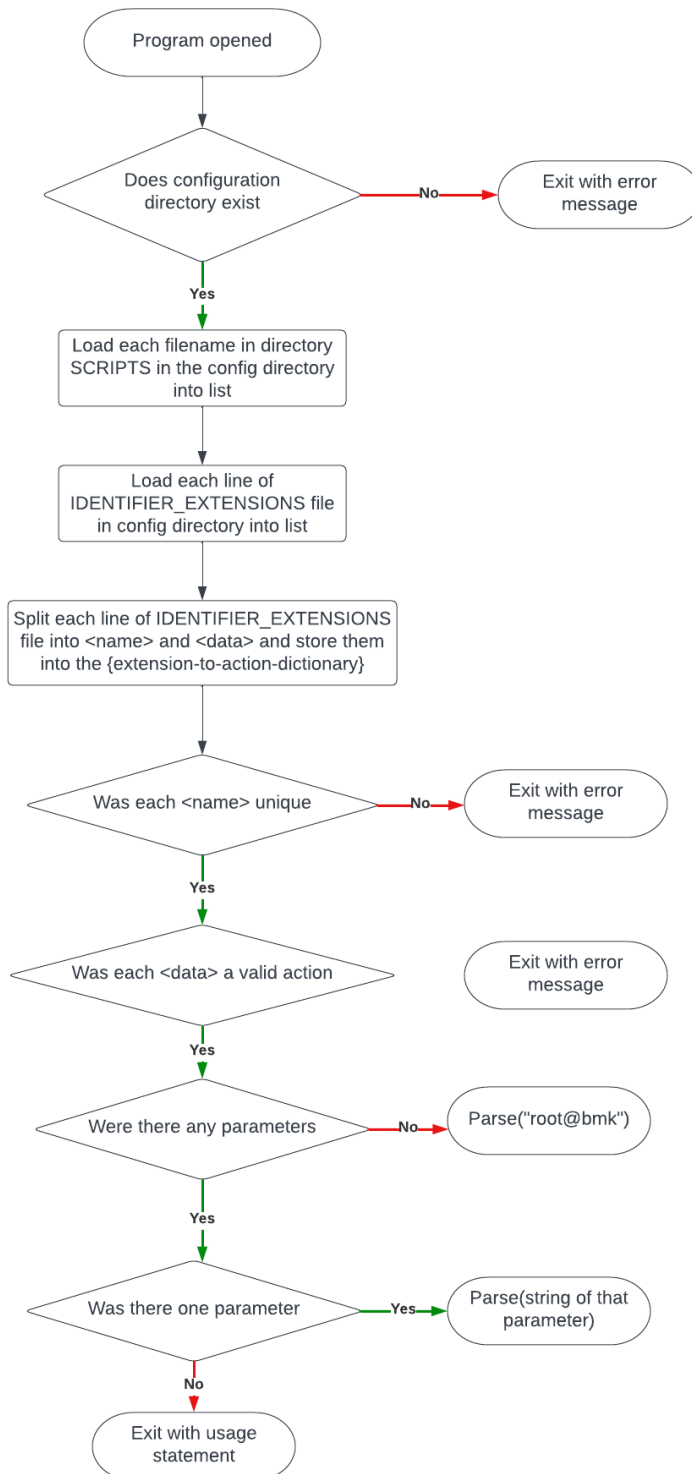
Therefore, I amended my idea to allow for users to give custom names `{identifier}|{script name}>{data}`. So, instead of writing "<https://www.youtube.com/watch?v=m3dRlTirgLU>", the user can instead add a bookmark called "`data-flow-diagrams|link>https://www.youtube.com/watch?v=m3dRlTirgLU`" with everything after the first '>' hidden in the menu if it is added as a bookmark, allowing for users to assign understandable identifiers for each bookmark.

I also wanted to create options that would directly interact with the menu when executed, such as bookmarks that would delete themselves as todo items or toggle between name and data as flashcards. Therefore I changed `{argument of script}|{script name}` to `{argument of script}{action delimiter}{action identifier}`, where `{action delimiter}` could be '|' indicating it is a script and also could be a ':' indicating it is a menu action and should be followed by an `{action identifier}` defined in the program. An example use case would be: "`how many levels of protein structure are their:flashcard>4`" in which the user can toggle between how many levels of "protein structure are their:flashcard" and "4."

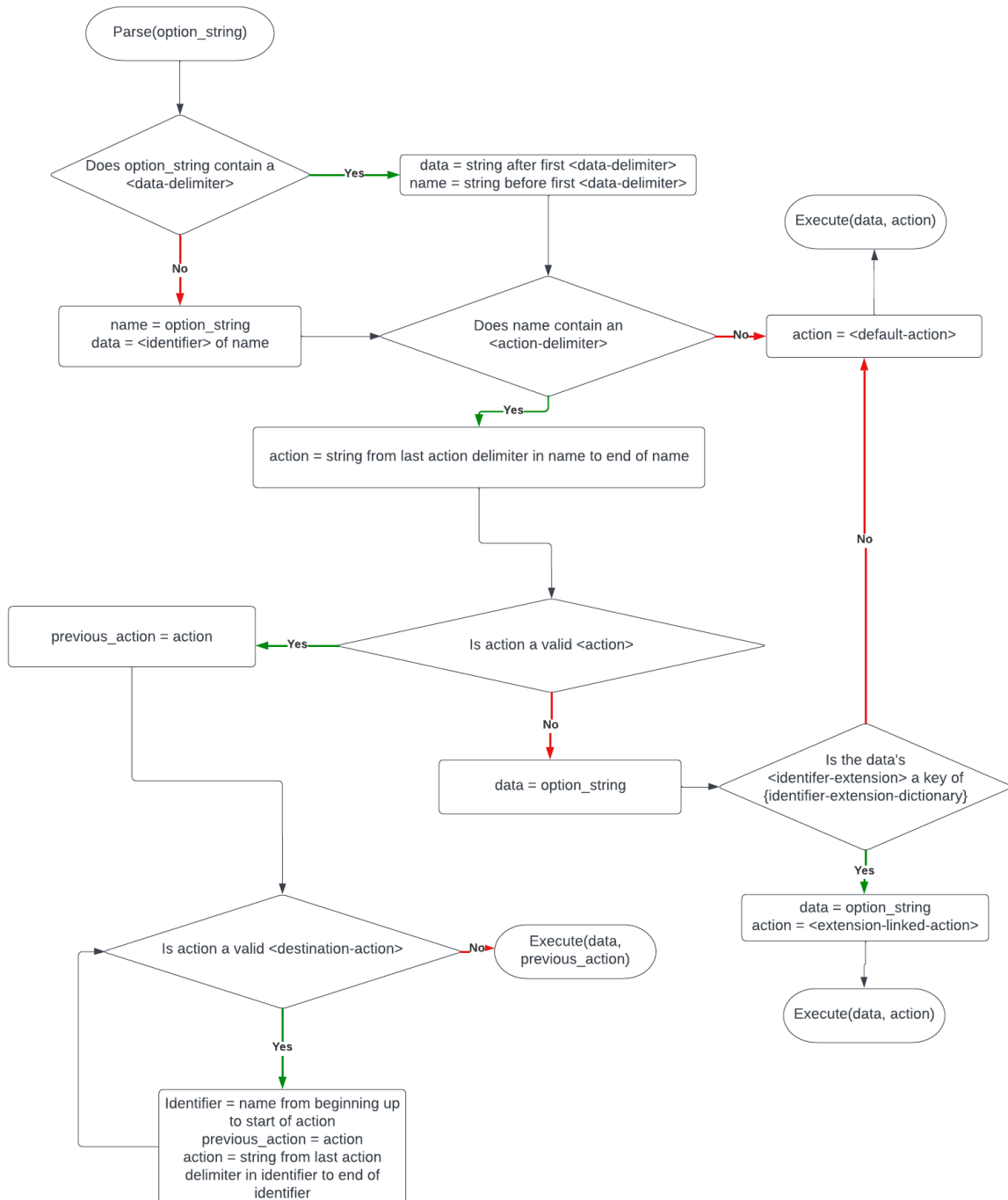
I added quite a few more features later on which I will define more formally in the next section, but this was my initial thought process when I created the syntax.

## Logic of program

### Start of program



### Parsing option string



## Configuration directory

Tree view of example config directory and descriptions of each subdirectory/file

- |— HISTORY #Stores the last n option strings executed
- |— IDENTIFIER\_EXTENSIONS #File that links identifier extensions to an action

```

|—— LOG #Log file
|—— OPTION_LISTS #Default location of file option lists
|   |—— characters
|   |—— google
|   |—— hello
|   |—— identifier-extensions
|   |—— nea
|   |—— root
|   |—— test
|   |—— todo
|—— SCRIPTS #User can create custom scripts here that will link to script actions
|   |—— app #Script to open an app
|   |—— link #Script to open a link
|   |—— music #Script to play an audio file
|   |—— search #Script to search
|   |—— text #Script to open a text file

```

\*Note everything after a '#' character is a comment

## Example IDENTIFIER\_EXTENSIONS file

com> link
txt> text
uk> link
org> link
> default
opus> music

## Menu interface design

### Layout

Title bar
Input box
Search option 1
Search option 2
Selected option - Search option 3
...

Status bar

## Title bar

First row of menu, contains information about location, permissions and mode of currently opened menu

### Format

```
{permission-start-bracket}{mode-character}{permission-end-bracket}
{location}<action>
```

### Example

[S] /home/user/Documents/file.pdf~option_string	This represents a user in an editable menu in search mode
---	---

## Input box

Contains text the user has inputted. In the default search mode, it searches through all the names of an option using an algorithm specified in the section - Search algorithm. In edit or insert mode, it contains the buffer of the text to update the menu.

## Search option

These contain the matched options from the current option list and input box that the user can select. If there are not enough matched options to fill all the columns, the columns without an option will remain empty with the selected option restricted to navigating there.

## Selected option

This option is the option that will be executed when the user presses Enter. A user can manually navigate the selected option using arrow keys. At the top, the up arrow key will do nothing, and likewise with at the bottom. When the selected option reaches the bottom of the visible menu but there are still more options below it, the option at the start row is shifted to the one below it.

## Status bar

Shows log messages in different colour coded categories

Category	Colour	Description
Error	Red	Error that has means that program cannot function properly
Warning	Yellow	User has invalid configuration or user

		has done something that may lead to an error
Info	White	General information to communicate to user in response to keypresses

## Modes of menu

### Edit mode

For editing current text at a manually changeable position.

### Insert mode

For inserting an option at a manually changeable position.

### Search mode

The default mode, for finding options and running them.

## Types of menu

### Bookmark menu

The main type of menu to be used. Each option's <data> is hidden initially but can be shown by the keypress CTRL-D. It is useful to store group related or common tasks together that have differing <action>s.

Opened by <destination-action> - "@bmk"

### Data menu

Each option's <name> is its <data>. This type of menu is for when you have a list of data that can all correspond to a specific action and already identifies itself meaningfully. This saves the user the time of having to name each option themselves.

Opened by <destination-action> - "@lst" or "@rlst"

### Directory menu

This menu is for exploring files. It is similar to a data menu except the <name> is the file name and the <data> is the absolute path of the file.

Opened by <destination-action> - "@dir" or "@rdir"

## Menu system example

(Using example config directory in section Config Directory)

name	data	action	What it does
------	------	--------	--------------

youtube.com	youtube.com	link	Opens the link youtube.com
google link	https://google.com	link	Opens the link https://google.com
school@lst:todo	school	@lst:todo	Opens option list called "school" located inside /path/to/config_direct ory/IDENTIFIER_EXTE NSIONS directory
chromium app	chromium	app	Opens an app called chromium
playlists@dir music>/path/to/music_dir	/path/to/music_dir	@dir music	Opens /path/to/music_dir where each option would be executed with  music
/path/to/music_playlists@dir@dir music	/path/to/music_dir	@dir@dir music	Opens /path/to/music_dir where each directory would be executed with @dir music such that each option executed will open a new "@dir" where each option would be executed with  music

## Menu 2

Option string to open menu: school@lst:todo

Pressing Enter should remove selected option or if there isn't none, add a new option

Geography homework - In my book
Monday's Physics lesson - revisit notes

## Menu 3

Option string to open menu: test@dir>Physics

Should list out files listed in Physics directory

## Menu 4

Option string to open menu: test@rdir>Physics

Same as Menu 3, but can't remove or edit anything

# Objectives

## General objectives

Specifies general objectives of my program.

1. Program can take in command line arguments
  - a. If there are no arguments, run the option string "root@bmk" opening the root menu
  - b. If one argument is provided, evaluate it as an option string
  - c. If an invalid amount of arguments are provided, print a usage statement
2. Render menu interface as shown in section - Menu Interface Design
  - a. Title renders correctly
  - b. Input box renders correctly
  - c. Search options render correctly
  - d. Status bar implemented correctly
3. Implement these interactions in search mode
  - a. Keybinds
    - i. Up - Navigate selected option to one after it if possible, shift whole view of option list up if new selected position is greater than the number of rows
    - ii. Down - Navigate selected option to one after it if possible, shift whole view of option list down if new selected position is greater than the number of rows
    - iii. CTRL-A - Add current input text at end of menu if it doesn't exist and menu is editable
    - iv. CTRL-R - Remove selected option if there is one and menu is editable
    - v. CTRL-E - Enter edit mode if menu is editable
    - vi. CTRL-I - Enter Insert mode if menu is editable
    - vii. CTRL-D - Toggle data
    - viii. CTRL-Q - Clear the input text
    - ix. Tab - Fill input text with selected option if possible
    - x. Printable key - Add character at cursor position if possible
    - xi. Backspace - Remove character before cursor position if possible
    - xii. Esc - Exit mode
    - xiii. Enter - Parse and execute option string of currently selected option if it exists, otherwise execute current input text as option string
    - xiv. CTRL-N - Execute current input text as option string
    - xv. CTRL-B - Navigate to previous opened menu in current session
    - xvi. CTRL-C - Copy from selected into clipboard
    - xvii. CTRL-V - Paste into input text
  - b. Events
    - i. Input text changes - Search using algorithm specified in objective 7
4. Implement these interactions in edit mode
  - a. Keybinds
    - i. Up - Navigate selected option to one before it if possible, shift whole view of option list up if selected position is greater than the number of rows



- ii. Down - Navigate selected option to one after it if possible, if at last row, scroll down, shift whole view of option list down if selected position is greater than the number of rows
    - iii. Enter - Replace selected with input text and flush backend and return to search mode
    - iv. Esc - Discard changes and return to search mode
    - v. Tab - Fill input text with first option name with the same starting characters as it if there is one
  - b. Events
    - i. Selected option changed - Set input text to name of currently selected option
    - ii. Input text changed - Set text at selected row to the contents of input text but don't flush any changes
    - iii. Enter edit mode - Set text at selected row to the contents of input text but don't flush any changes
- 5. Implement this interface when in insert mode
  - a. Keybinds
    - i. Up - Navigate selected option to one after it if possible, shift whole view of option list up if new selected position is greater than the number of rows
    - ii. Down - Navigate selected option to one after it if possible, shift whole view of option list down if new selected position is greater than the number of rows
    - iii. Enter - Insert at position of option to insert
    - iv. Esc - Discard changes and return to search mode
  - b. Events
    - i. Enter insert mode - Set text at selected row to the contents of input text but don't flush any changes
- 6. Changes to option list in menu are mirrored in where they were loaded from so that changes to menu are saved
  - a. User can load option list from directories where each file name corresponds to an option
    - i. User can edit the file name within the menu
    - ii. User can add files within the menu
    - iii. User can remove files within the file
  - b. User can load menus from files where each line of the file corresponds to an option
    - i. User can edit file lines within the menu
    - ii. User can add file lines within the menu
    - iii. User can remove file lines within the file
  - c. The specifics of how they are edited depends on the type of menu that has been opened
- 7. Search results must be ordered by priority
  - a. At the top there will be exact matches, where input text exactly matches name of the option
  - b. Next, prefix matches, where first character of each token is found in the option string
  - c. Finally, substring matches, where each token is a substring in the option string
  - d. If matches are the same type, order of original list from which options are searched should be retained
- 8. Configuration directory must be loaded at the start of the program

- a. Checks if configuration directory exists at path specified at environment variable MY\_MENU\_CONFIG, otherwise looks in default location of \$HOME/.config/my\_menu, if not, an error is thrown
- b. Checks if all files and directories that are supposed to be in directory are there, if not, error is thrown

## Extension objectives

- 9. History file
  - a. All option strings executed are appended to end of history file
  - b. If number of lines exceeds a set amount, the oldest option strings (the ones at the start) will be removed
- 10. Allow for composite option lists
  - a. Multiple option lists can be appended together so that program can search through multiple option lists
  - b. Editing a specific option in this menu will only edit that menu
  - c. This can be done by adding to the syntax a new type of directory action that expands any options with directory actions it finds within it
- 11. Develop this program to be cross platform
  - a. Work on both Windows and Linux and operating systems
- 12. Implement mouse support
  - a. User can execute an option by left clicking on it
  - b. User can toggle data of an option by right clicking on it

## Objectives about program specifics

Specifies in detail objectives about how an option string should be evaluated and executed and how details about different types of menus

- 13. Can implicitly assign an action to an option string when one isn't assigned explicitly
  - a. If it is a path to a directory, open it with the action "@dir"
  - b. If it has an extension, e.g. ".com" or ".txt", it will look at the user configuration file linking extensions to actions to assign an action
  - c. If the user configuration file linking extensions to actions has an extension that is an empty string, the action that links with that will be assigned as the default action if the option string falls through to here
  - d. Otherwise, execute the action with ":nothing", which should notify the user that nothing has been done because no default action has been found through the status bar
- 14. Identifies longest valid action and uses it as action of the option string when an option string is assigned explicitly

## Examples

option string	action
chrome app	app
choose_playlist>/path/to/playlists	{default-action}
choose_song@rdir@rdir music>/path/to/playlists	@rdir@rdir music

15. Data is implicitly assigned as data if no data is explicitly given with a data delimiter

#### Examples

option string	action	Condition
hello world	search	Line "> search" in IDENTIFIER_EXTENSIONS config file
google.com	link	Line "com> link" in IDENTIFIER_EXTENSIONS config file

16. Menu actions can manipulate the menu in their respective ways
- ":todo" and ":tmp" - Removes it if it is selected, if it is the input, it adds it to the menu
  - ":flashcard", ":setting" and ":toggle" - Toggles data on and off if selected, if it is the input, it adds to the menu
  - ":scratchpad" - Toggles data on and goes into edit mode if selected, if it is the input, it adds to the menu
17. Destination actions can open different menus
- "@bmk" - Opens bookmark menu
    - If data refers to a file path, load that path, else open/create and open file in config directory OPTION\_LISTS and set that to location of menu
    - Load the file into the menu, with each line representing an option string to add
    - Option string split up into name and data when added to the menu, the data is everything after first '>' and the name is everything before
    - If no '>' in string, automatically add one at end
    - Data is hidden in display of menu
    - Each change of menu is mirrored in file so that changes are saved
  - "@lst" - Opens data menu
    - If data to open this menu refers to a file path, load that path, else open/create and open file in config directory OPTION\_LISTS and set that to location of menu
    - Load the file into the menu, with each line representing data
    - Whole string is shown in display of menu
    - By default, opened as "~option\_string" meaning each line is evaluated as an option string
    - Each change of menu is mirrored in file so that changes are saved
  - "@rlst"
    - Same as "@lst" but one can't add/remove/edit options
  - "@dir"
    - Data to open this menu should refer to a directory path
    - Load file into menu, with each filename being displayed in menu
    - Each option's data is the the file's absolute path
    - Each change of menu is mirrored in directory so that changes are saved
  - "@sdir"
    - Same as "@dir" but files created are executable

- f. "@dir"
  - i. Same as "@dir" but cannot edit
- 18. Script actions run scripts in the SCRIPTS directory in the config directory with the same name as the action identifier.
- 19. Can assign actions to execute on a destination action by adding it between destination action and first data delimiter or end of string

#### Example

option string	What it does
nea@bmk:todo	Opens option list nea with each option's data being executed as a todo item

## Alternative solutions and justification of chosen solution

### Alternative solutions

1. A document editor like Microsoft Word or Google Docs where you can directly put in images, links, etc
  - a. It is not feasible to integrate all different types of data into a document editor due to time constraints and my ability.
  - b. Must put whole of information there, so harder to look for specific information as you can't search for a specific image for example
  - c. Less convenient to navigate between documents without a search feature
  - d. Less minimal, too much information may be detrimental to the user's focus
2. A normal app launcher without the extra syntax
  - a. Does not solve core issue of aggregating different types of computer tasks together
  - b. May want to rename apps to more memorable names
  - c. Does not allow user to add bookmarks that provide direct access to a task except to open apps

### Chosen solution

My solution allows users to group together actions similar to how humans group things. The user will not be restricted by the fact that they don't execute the same way so they can't be put together, as they can specify that in the option. It also provides a convenient centralised access to commonly used tools like a file browser and todo lists, improving the user's productivity. The ability to search using text greatly improves the time for the user to find information, and the fact that the user can give custom names only enhances that. Also, it is a lot more lightweight than opening a document so it can be used often to startup programs. Finally, while the user having to make their own scripts may be a detriment to the simplicity of the program, it gives more freedom to the user to expand with their own scripts, allowing them to be more productive in the long term.

## Acceptable limitations

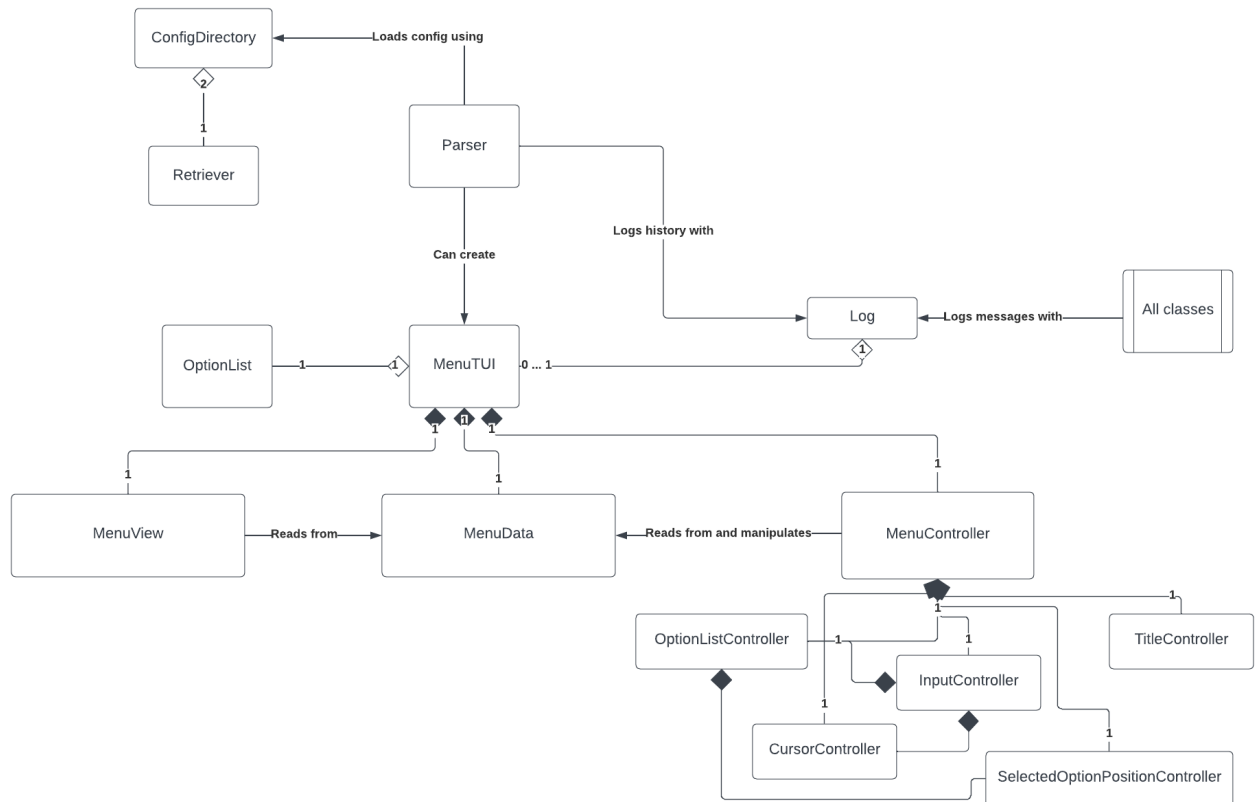
- A prerequisite for the user to be able use this tool effectively is a basic knowledge of scripting. I will note that it should be possible for someone to design a configuration directory with scripts such that it will provide a working system for almost all users using the operating system that configuration directory was designed for. However, this will not be part of my project.
- The program is designed to be opened and closed by the user as opposed to be opened at startup. Hence, keybinds to open it must be handled by the user if they want to use it as a launcher.
- The program will be able to run on Linux systems, and potentially on Windows systems depending on if I do the extension objective
- The program will not check if the user's scripts are acceptable i.e. takes in one argument

## Documented design

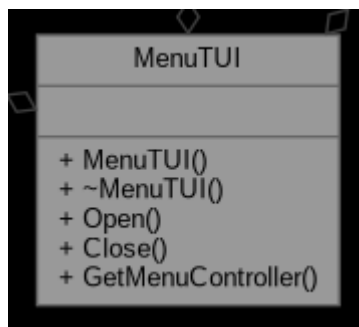
I have used different diagrams, pseudocode and written communication to model and explain key areas of my code. Note that in my UML class diagrams, I have not included parameters and types of functions as I think header files cover a similar purpose. Also, for the same reason, I have not gone in depth about each function and class as I have done so in the header files as comments.

## Class relationships

This diagram shows the relationships between different classes using UML relationship arrows. It does not show any inherited classes otherwise there would be too many. I will specify show design of inherited classes when I go into specific classes.



## MenuTUI Class



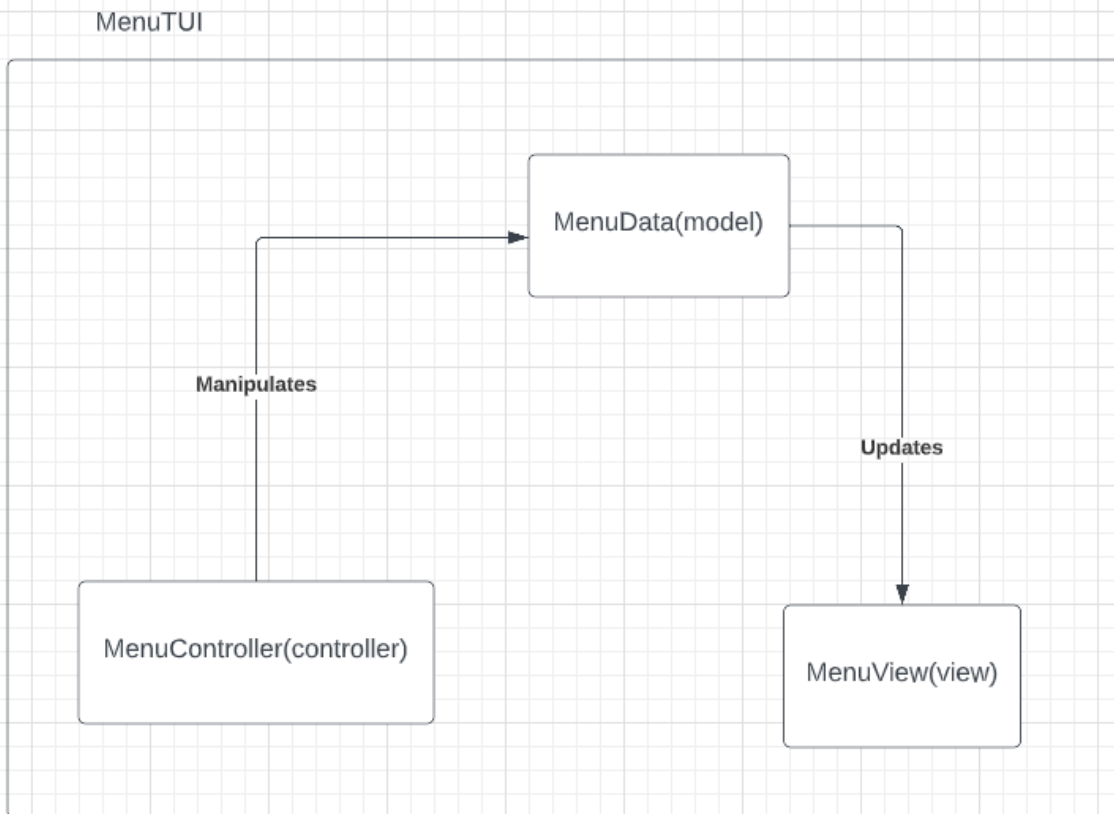
## MVC design pattern

For the MenuTUI class, I used the MVC design pattern. This has several advantages over using a single menu class:

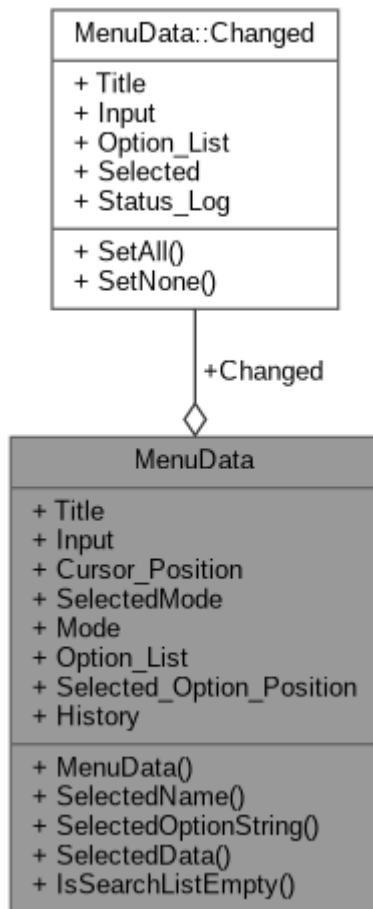
1. **Separation of concerns:** The MVC pattern separates the concerns of an application into three distinct components, which makes the code easier to understand, maintain, and modify. This separation ensures that changes to one component do not affect the others.
2. **Improved scalability:** The MVC pattern helps improve the scalability of an application by providing a clear separation of concerns. This makes it easier to add new features or modify existing ones without affecting other parts of the application.
3. **Reusability:** The MVC pattern promotes code reusability by separating the application into distinct components. This makes it easier to reuse code in other projects or applications.

4. Testability: The MVC pattern makes it easier to test an application's individual components, as they are separated into distinct parts. This makes it easier to create unit tests and ensure that the application functions as expected.

The MenuTUI class encapsulates all components of a menu. The associations of the classes are represented in the diagram below.



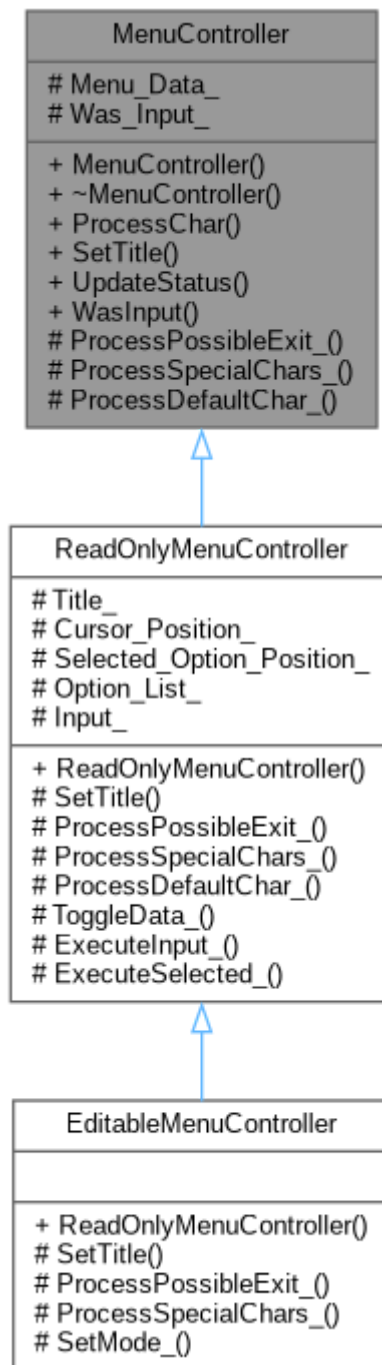
## MenuData



Read by class MenuView to display the menu and read from and updated by MenuController class to change class. It contains a struct called Changed which is updated by MenuController class to indicate which parts of the menu need updating and reset after a keypress has been processed and changes have been displayed.

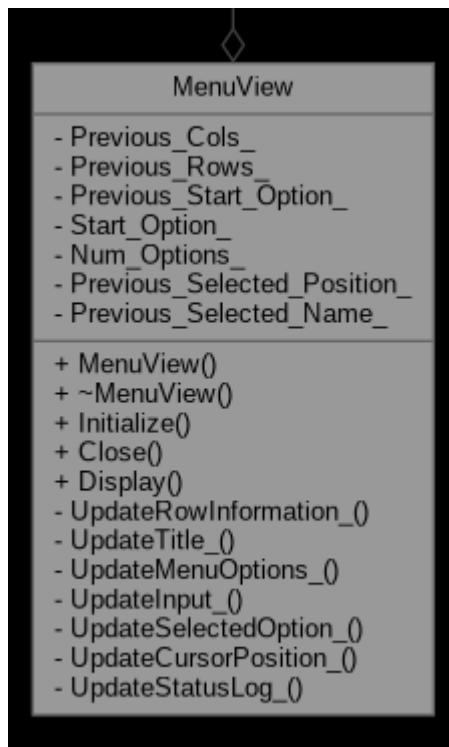


## MenuController



The MenuController class handles user keystrokes and updates MenuData accordingly. It contains various sub-components that manage specific parts of MenuData relating to specific components of the menu, i.e. InputController controls the input box. This class is abstract, and different permission modes inherit from it.

## MenuView



The `MenuView` class handles rendering the information in `MenuTUI` onto a menu using the `Ncurses` library. It has many update functions that relate to different parts of the menu, which will each update if `MenuController` has updated the respective `MenuData::Changed` struct.

## History stack

History is a static member variable of `MenuData`. It is a stack that stores the past opened menus, storing their location and the action to open it, which is all the information needed to store a menu. The top of the stack represents the most recent menu opened, while the bottom of the stack represents the root menu, the menu that was initially opened. The stack is modified after these events.

Menu is opened: Called by constructor of `MenuTUI`

`History.push()` is executed to make current menu most recent

CTRL-B pressed to go back in history: Called in `ReadOnlyMenuController::ProcessChar`

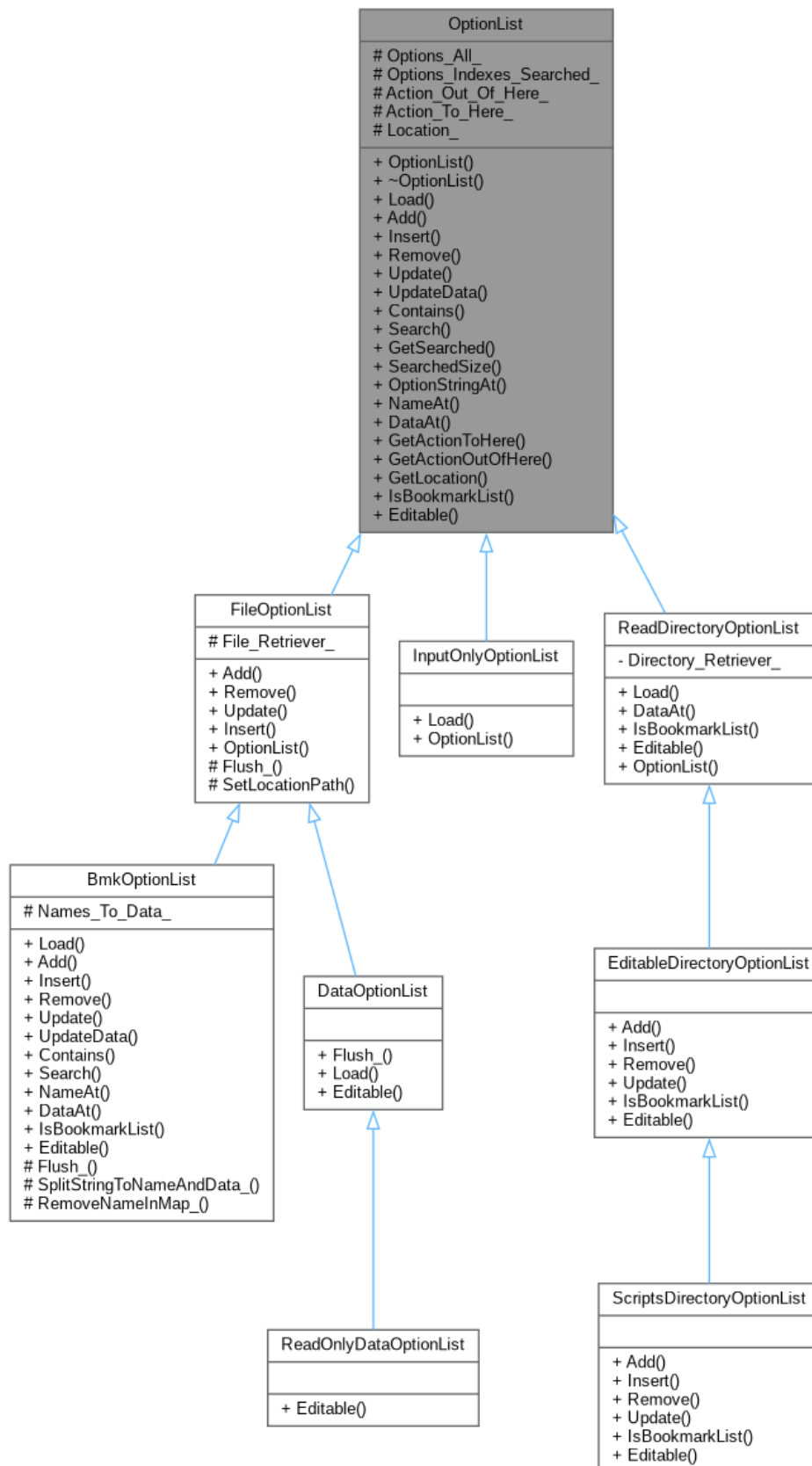
If only one menu in stack, nothing is done as they are in the root menu

`History.pop()` is called to remove current menu from the stack

`History.peak()` is called and program saves menu to go back to in stack

`History.pop()` is called again - As after opening the menu again that menu will automatically be repushed into the stack.

## OptionList class



This is an abstract class which defines the list of options that will be searched. With an object of the **OptionList** class, a **MenuTUI** can be loaded. It defines virtual functions

that define basic interactions with a list of strings each corresponding to an option string that can be executed with the `std::vector Options_All`. Derived classes define specific functionality of interacting with a type of menu and also implement the updating of the backend(files names or lines of files) of options that are changed.

## Search algorithm

### Tokenized matching algorithm

Tokenization is the process of converting a sequence of characters (such as in a computer program or web page) into a sequence of lexical tokens (strings with an assigned and thus identified meaning).

When humans try to recall something, they often use partial cues or keywords to trigger memories rather than recalling the exact information. Therefore, my algorithm splits the input text to search into tokens. The tokens are then matched in the search array in different priorities

1. Exact match - Input text exactly matches an item in the search array
2. Prefix match - Every first character of each token is found in the search array
3. Substring array - Each token is a substring of the search array

Otherwise, the order of the original search array is preserved. Using the following algorithm, it adds the indexes at SearchArray of all matched entries. This is as position of matched entry also needs to be preserved for operations such as insert, where you insert one before the currently selected option.

### Pseudocode

```
MatchedArray Search(String Input, Array SearchArray):
    IF (Input = "") RETURN [1 -> SearchArray.Size]
    List ExactMatches <= new List
    List PrefixMatches <= new List
    List SubstringMatches <= new List

    List InputTokens <= Input.Split(AnyOf(" .-_"))

    FOR INT i = 1 -> SearchArray.Size:
        bool AllTokensFound <= True;

        FOREACH Token In InputTokens:
            IF (NOT(SearchArray[i].ContainsSubstring(String)):
                AllTokensFound <= False
                BREAK

    IF (AllTokensFound):
        IF (Input = String):
            ExactMatches.Add(i);
        ELSE IF (SearchArray[i].Contains(String[0]) == 0:
            PrefixMatches.Add(i);
        ELSE
            SubstringMatches.Add(i);
```

RETURN ExactMatches + PrefixMatches + SubstringMatches

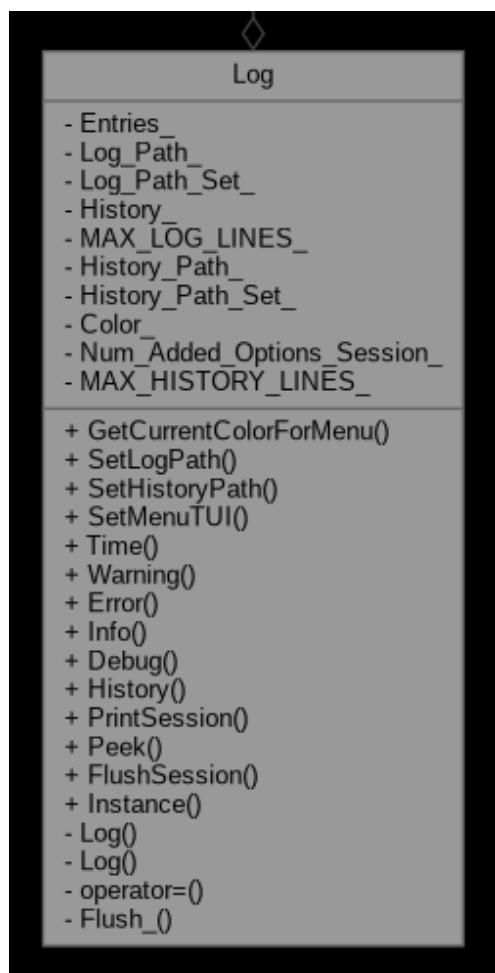
## FileOptionList

Abstract class derived from OptionList that adds manipulation of files that mirrors the editing of the current option list. Does not add the loading of files as that has a different implementation for classes that derive from it.

## InputOnlyOptionList

When loading a file fails, this is opened. It can take in inputs but all the functions that interact with an OptionList does nothing.

## Log class



## Singleton design pattern

The log class was a singleton class globally accessible to all classes. This is because all classes could use it and there should only be one instance of it, so it seemed sensible. It also allows for possible inheritance for different implementations of the log class.

## Fixed length queue

This class used a fixed length queue which was a small custom class I created that derived from the `std::queue` the C++ library. I used a fixed length queue for my command history class, which would store the used commands

## Implementation

It can be used as follows:

```
my::log.{LogType}() << "text goes here" << std::endl;
```

Where {LogType} is one of the functions Error, Warning, Info, Debug each representing a different type of log. It adds the corresponding prefix before the text, for example "Error: " for Errors. `std::endl` indicates to flush the text into the back of the deque saving the logs.

Or as follows:

```
my::log.History("text goes here");
```

Which pushes the text into the back of the deque saving the past commands.

## Builder design pattern

The functions `SetHistoryPath(path)`, `SetLogPath(path)` and `SetMenuTUI(path)` will set what they respectively set. After they have been set, the Log class will utilise them. If they have not been set, the log will not utilise them. In other words, after `SetLogPath` function has been run, the Log class will flush the logs to the path specified after the program has finished. After the `SetHistoryPath` function has been run, it will similarly flush the past commands into the history path specified. Finally, if `SetMenuTUI` function has been triggered, it will update the status of the MenuTUI using `MenuTUI::MenuController::UpdateStatus`.

## History and log data structure

There are two deques in the Log class, one for storing past option strings executed and one for storing logs. It is a deque, in case I want to implement a feature that removes most recent logs after the user has read them in that status log. The back of the deque represents the most recent logs/option strings

## Flushing history and data

`FlushSession` is a function binded to the event `atexit`.

At the end of the program, history and data are appended to their respective files if they have been set. To ensure the size of each file is limited, if the number of lines of the file exceeds a set amount, those lines are removed. Therefore those files behave similar to queues.

## Pseudocode

```

Flush(path, entries, max_lines):
  file <- OPEN_FILE(path, APPEND)
  FOR entry: entries.START -> entries.END:
    file.APPEND(entry)

  line_count <- file.COUNT_LINES()
  file.CLOSE()
  excess_lines = line_count - max_lines
  IF excess_lines > 0:
    # Can't directly remove lines so create new file and write first max_lines
    infile <- OPEN_FILE(path, READ)
    outfile <- OPEN_FILE("temp.txt", WRITE)

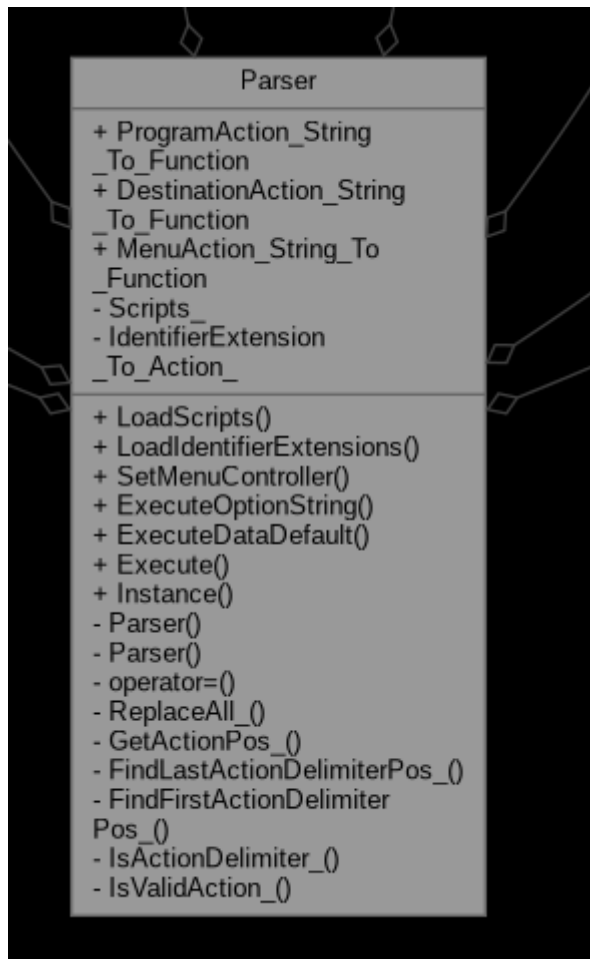
    FOR i: 1 -> excess_lines:
      line = infile.GETLINE() #Lines don't have random access

    FOR i: 1 -> max_lines:
      line = infile.GETLINE()
      outfile.APPENDLINE()

  #Replace file at path with temporary file
  REMOVE_FILE(path)
  RENAME_FILE("temp.txt", path)

```

## Parser Class



The parser class deals with the parsing and execution of strings.

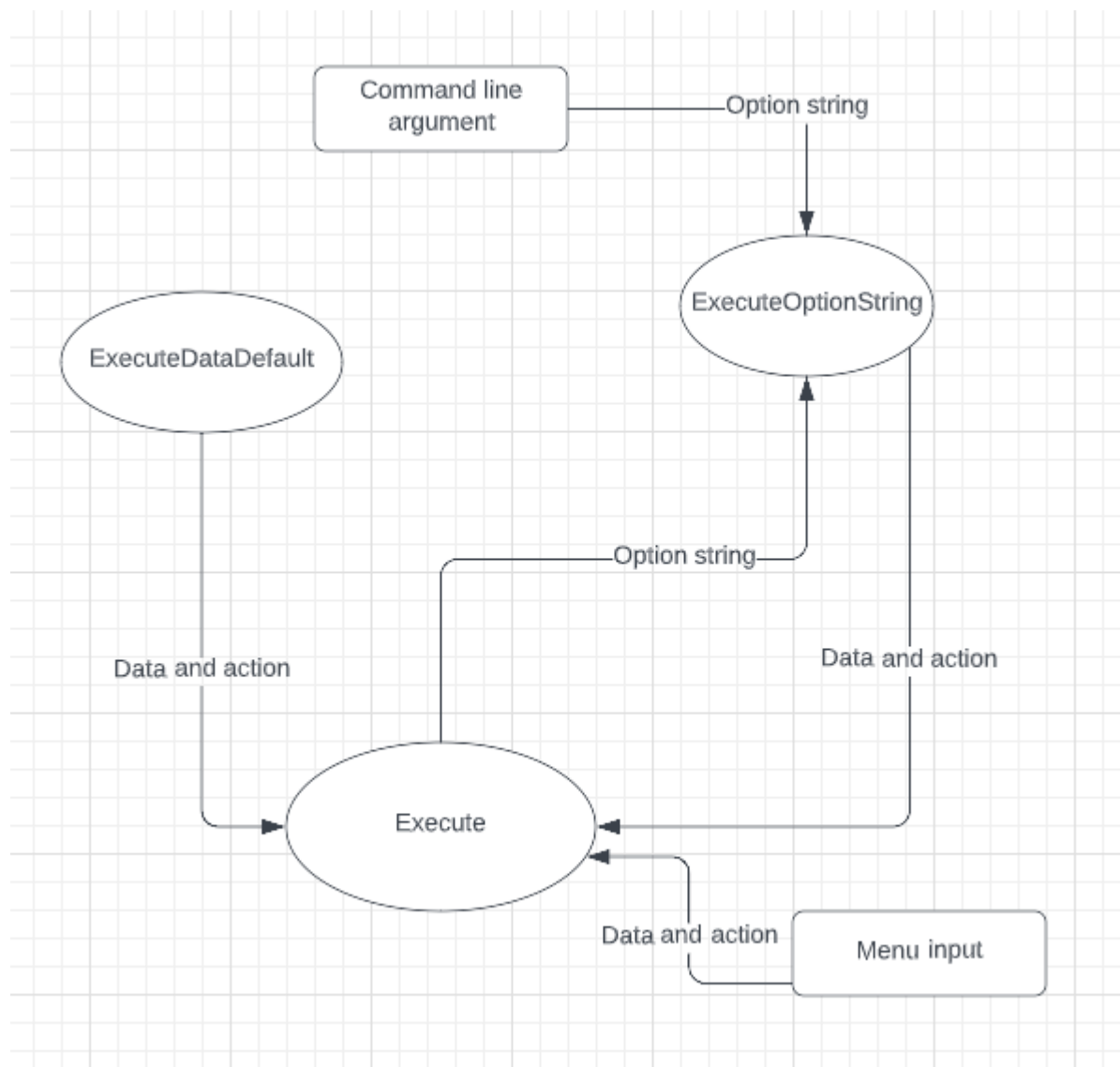
## Initialization

Before the Parser class is used, the LoadScripts and LoadIdentifierExtensions functions must be called. They load the filenames of the script directory into the Scripts\_ std::set and the user sets identifier extensions into the IdentifierExtension\_To\_Action\_ std::unordered\_map for use by the parser functions for execution.

## Data flow diagram of execution

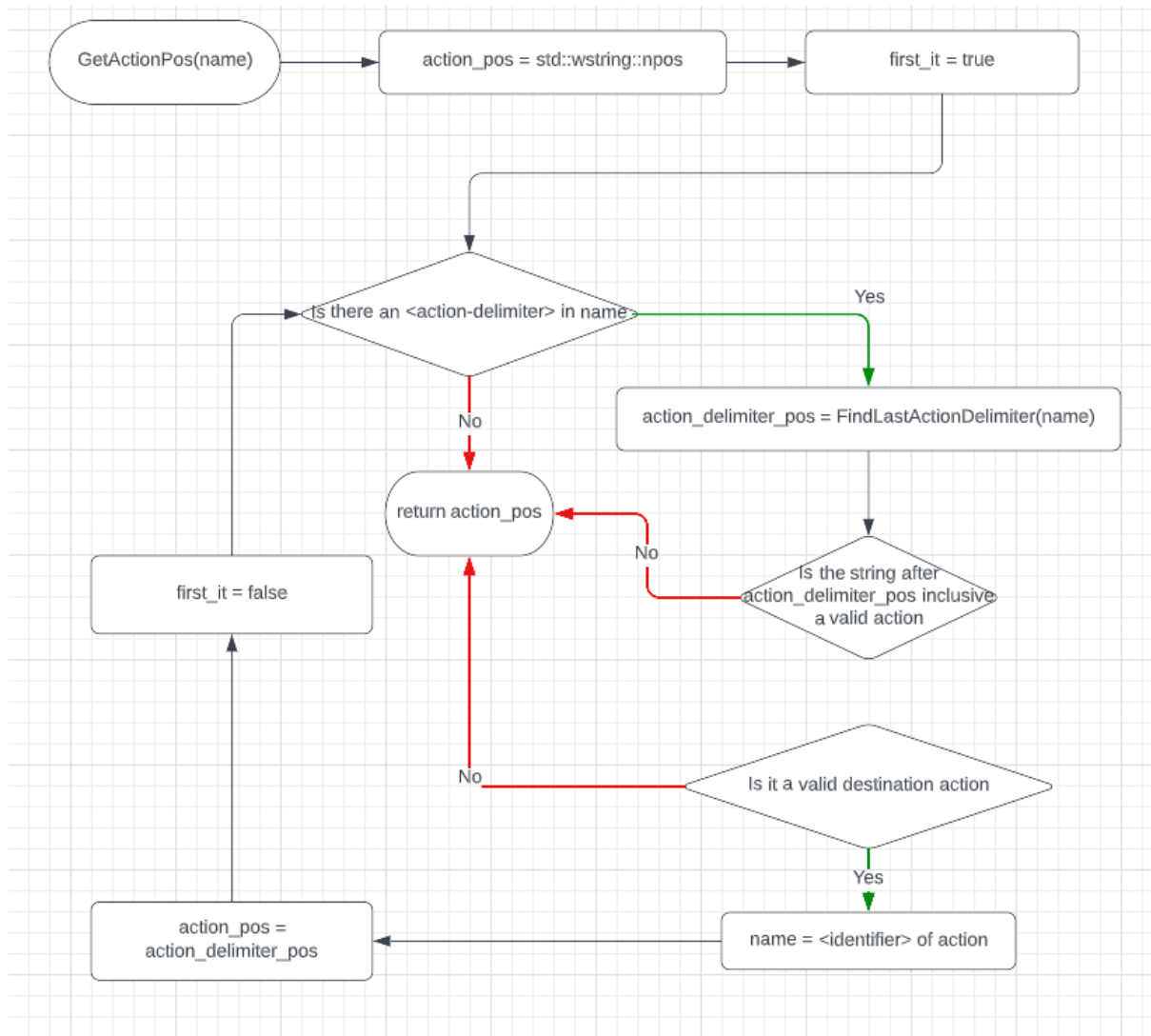
This is the data flow through different processes in the execution of a string.





## Finding position of action using recursion

Starting from the back, the algorithm must find the position of the longest valid action. It must handle the fact that destination actions can prepend other actions, there are no action delimiters and that there are action delimiters that are just part of the data, and therefore don't correspond to an action. The simplest way of explaining it is that it finds the last action delimiter, sees if the action defined by that action delimiter is valid, and continues back until it finds one that is invalid. When it finds one that is invalid or doesn't find one, it returns the previously found one as the position of action in an option string. To do this, I use a recursive algorithm represented with the following flowchart:



## String to function dictionaries

I have defined several dictionaries/`std::unordered_maps` to connect their action identifiers to their respective hard-coded functions.

- `MenuAction_String_To_Function` - Takes in a `MenuController*` of the current function and parses keypresses into them to run a function manipulating the function
- `DestinationAction_String_To_Function` - Takes in the action of the option string executed to create the menu, the action that will be executed by default at the menu and the location of the menu to create a `MenuTUI` object.
- `ProgramAction_String_To_Function` - Takes in data of option string and does something with it

This allows me to easily connect hard coded actions to their respective action identifiers during execution without having to use comparison statements. For example: `MenuTUI menu_tui = MenuTUI(DestinationAction_String_To_Function.at(destination_action_identifier)(action_at_destination, action, processed_data));`

It also allows me to easily check if the action identifier exists for the corresponding action delimiter, shown by the following pseudocode.

### Pseudocode

```
IsValidAction(action_del, action_identifier):
    SWITCH (action_del):
        CASE (DestinationAction::Delimiter):
            RETURN
            DestinationAction_String_To_Function.contains(action_identifier)
            BREAK
        CASE (ProgramAction::Delimiter):
            RETURN
            ProgramAction_String_To_Function.contains(action_identifier)
            BREAK;
        CASE (ScriptAction::Delimiter):
            RETURN Scripts_.contains(action_identifier)
            BREAK;
        CASE (MenuAction::Delimiter):
            RETURN
            MenuAction_String_To_Function.contains(action_identifier)
            BREAK;
```

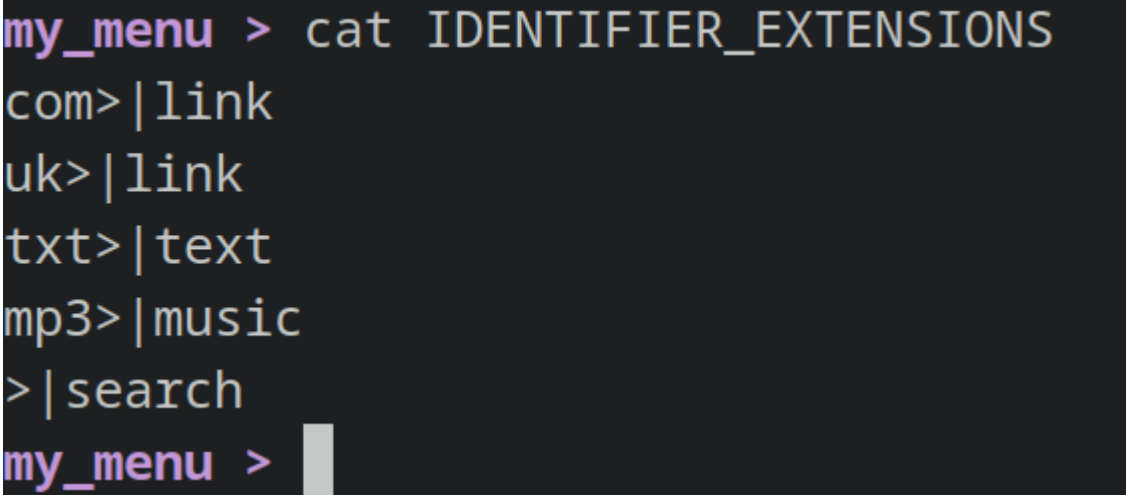
## Testing

In testing, I try and input at least one normal, boundary and erroneous data when applicable.

### Configuration directory

This is the configuration directory I am using for testing. I will reset the configuration files to this state each time I run a test.

#### IDENTIFIER\_EXTENSIONS file



```
my_menu > cat IDENTIFIER_EXTENSIONS
com>|link
uk>|link
txt>|text
mp3>|music
>|search
my_menu >
```

## SCRIPTS directory

Table describing what each script does

Script	What it does
link	Opens data as link with web browser
app	Opens data as app
text	Opens text file with text editor with data as path to text file
search	Opens web browser and searches with data as search
music	Opens and plays audio file

Evidence and contents of scripts

```
SCRIPTS > ls
app  link  music  search  text
```

```
SCRIPTS > cat app
#!/bin/bash

$1 & disown
```

```
SCRIPTS > cat link
#!/bin/bash

link=$1
if [[ ! $link =~ ^(https?|ftp|file):// ]]; then
    link="https://$link"
fi

chromium --new-window --app=$link & disown
```

```
SCRIPTS > cat music
#!/bin/bash

vlc "$1" & disown
```

```
SCRIPTS > cat search
#!/bin/bash

engine="https://google.com/search?q=%s"
protocol='^(https?|ftp|mailto|about|file):///?'

input=$1
checkurl() {
    [ ${#input} -lt 4 ] && return 1
    echo "$input" | grep -Z ' ' && return 1
    echo "$input" | grep -EiZ "$protocol" && return 0
    echo "$input" | grep -FZ '..' && return 1
    prepath=$(echo "$input" | sed 's/(\/|#|\?).*//')
    echo "$prepath" | grep -FvZ '.' && return 1
    echo "$prepath" | grep -EZ '^([[:alnum:]]~_-[:~\.\?]){1,3}' && return 0
}

if checkurl
then
    echo "$input" | grep -EivZ "$protocol" && input="https://$input"
else
    input=$(echo "$engine" | sed "s/%s/${input% 0a}/;s/[[:space:]]/+/g")
fi

$BROWSER "$input" & disown
```

```
SCRIPTS > cat text
#!/bin/bash

alacritty -e nvim $1 & disown
```

## Testing environment

I am testing on Ubuntu Linux.

./program.program is the path to my executable.

## Testing core objectives

### Objective: 1.a

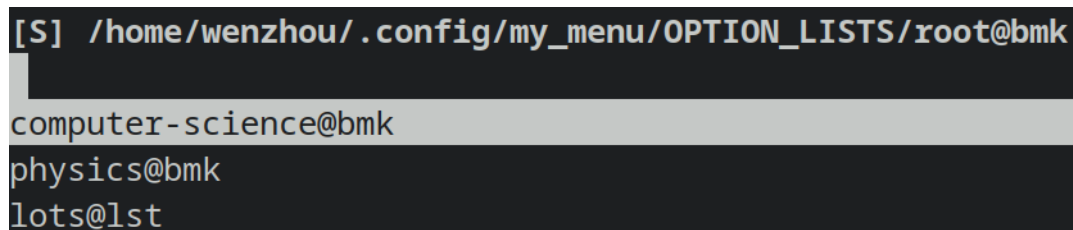
#### Test

Executed “./program.program” in a terminal with the bash shell

#### Expected result

Program is opened with path to root@bmk option list in the title bar

#### Actual result



```
[S] /home/wenzhou/.config/my_menu/OPTION_LISTS/root@bmk  
computer-science@bmk  
physics@bmk  
lots@lst
```

As expected

### Objective: 1.b

#### Test 1

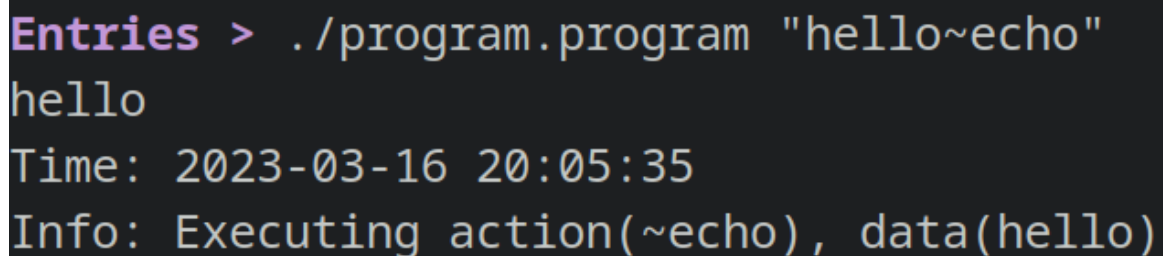
Executed “./program.program ‘hello~echo’” in a terminal with the bash shell

#### Expected result

hello is echoed back in command line

#### Actual result

As expected



```
Entries > ./program.program "hello~echo"  
hello  
Time: 2023-03-16 20:05:35  
Info: Executing action(~echo), data(hello)
```

### Objective: 1.c

#### Test 1

Executed “./program.program ‘hello~echo’” in a terminal with the bash shell

Expected result

Returns usage statement

Actual result

```
Entries > ./program.program param_1 param_2
Time: 2023-03-16 20:15:00
Error (1) : Usage - ./program.program option_string
```

As expected

Objective: 2

Test 1

Executed “./program.program ‘root@bmk’” in a terminal with the bash shell

Expected result

As described in section - Menu Interface

Actual result

```
[S] /home/wenzhou/.config/my_menu/OPTION_LISTS/root@bmk
]
computer-science@bmk
physics@bmk
lots@lst

Info: Loaded option list successfully
```

As expected

Objective: 3.a.i + 4.a.i + 5.a.i

Test 1

Pressing up arrow at top of menu

Expected result

Nothing happens

## Actual result

### Before

```
[5] /home/wenzhou/.config/ry_menu/OPTION_LISTS/root@bash
]
Computer science@en
Physics@en
List@list

INFO: loaded option list successfully
```

### After

```
[5] /home/wenzhou/.config/ry_menu/OPTION_LISTS/root@bash
]
Computer science@en
Physics@en
List@list

INFO: loaded option list successfully
```

### As expected

## Test 2

Pressing down arrow when the selected option is in the second row

## Expected result

Goes to top row

## Actual result

### Before

```
[5] /home/wenzhou/.config/ry_menu/OPTION_LISTS/root@bash
]
Computer science@en
Physics@en
List@list

INFO: loaded option list successfully
```

### After

```
[5] /home/wenzhou/.config/ry_menu/OPTION_LISTS/root@bash
]
Computer science@en
Physics@en
List@list

INFO: loaded option list successfully
```

### As expected



### Test 3

Press up arrow when selected position is at least two below bottom of total rows

Expected result

Whole view of rows shifts up by one

Actual result

Before

```

15] /home/wenzhou/.config/xy_menu/OPTION_LISTS/lotuslist
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
Info: Loaded option list successfully

```

After

```

15] /home/wenzhou/.config/xy_menu/OPTION_LISTS/lotuslist
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
Info: Loaded option list successfully

```

As expected

Objective: 3.a.ii + 4.a.ii + 5.a.ii

### Test 1

Pressing down arrow at bottom of menu

Expected result

Nothing happens

Actual result

As expected

### Test 2

Pressing down arrow when the selected option is in the second row

Expected result

Goes to third row

Actual result

As expected

### Test 3

Press down arrow when selected position is at least two below bottom of total rows

Expected result

Whole view of rows shifts down by one

Actual result

As expected

### Objective: 3.a.iii

#### Test

Wrote hello in input in "root@bmk" and pressed CTRL-A and the cleared input using CTRL-Q

Expected result

New row called hello at end

Actual result

Before

```
[5] /home/wenzhou/.config/my_menu/OPTION_LISTS/root@bmk
computer-science@bmk
physics@bmk
lists@list
Info: Removed string: hello
```

After

```
[5] /home/wenzhou/.config/my_menu/OPTION_LISTS/root@bmk
computer-science@bmk
physics@bmk
lists@list
hello
Info: Added string: hello
```

As expected

### Objective: 3.a.iv

#### Test

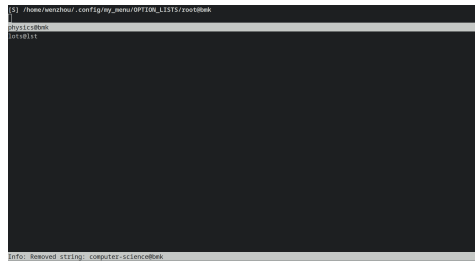
Pressed CTRL-R after opening "menu@bmk"

Expected result

computer-science@bmk gone from option list

Actual result

After



As expected

## Objective: 3.a.v

Test

Pressed CTRL-E after opening "menu@bmk"

Expected result

Input set to selected and [E] at start of title bar

Actual result

As expected

## Objective: 3.a.vi

Test

Pressed CTRL-I after opening "menu@bmk"

Expected result

Input cleared and [I] at start of title bar

Actual result

As expected

## Objective: 3.a.vii

Test 1

Pressed CTRL-D after opening "menu@bmk"

Expected result

Selected row to be empty

Actual result

As expected

Test 2

Pressed CTRL-D twice after opening "menu@bmk"

Expected result

First row to go back to what it was

Actual result

As expected

Objective: 3.a.viii

Test

Wrote hello and then pressed CTRL-Q after opening "menu@bmk"

Expected result

First row to be empty

Actual result

As expected

Objective: 3.a.viii

Test 1

Pressed tab after opening "menu@bmk"

Expected result

Input text to be filled

Actual result

As expected

Test 2

Pressed tab after opening "menu@bmk" and writing "sldkjsf"

Expected result

Nothing happens

Actual result

As expected

## Objective: 3.a.x

### Test 1

Pressed keys "abcdef"

Expected result

"abcdef" to be written to input box and cursor to be set one character the right of f

Actual result

As expected

### Test 2

Pressed keys "abcdef", pressed left arrow twice, pressed a

Expected result

"abcdaef" to be written to input box and cursor to be set one character the right of f

Actual result

As expected

## Objective: 3.a.xii

### Test

Pressed ESC after opening "menu@bmk"

Expected result

Exits program

Actual result

As expected

## Objective: 3.a.xiii

### Test 1

Pressed enter after opening "menu@bmk"

Expected result

Option list to switch to computer-science@bmk

Actual result

As expected

## Objective: 3.a.xiv

### Test 1

Pressed enter after opening and writing hello "revision@lst:todo"

Expected result

Search script to run on "hello"

Actual result

As expected

## Objective: 3.a.xv

Test 1

Pressed CTRL-B after opening "menu@bmk" directly from terminal

Expected result

Nothing happens

Actual result

As expected

Test 2

Pressed CTRL-B after opening "menu@bmk" directly from terminal and then opening "computer-science@bmk"

Expected result

Goes back to menu@bmk

Actual result

As expected

## Objective: 3.a.xvi and 2.a.xvii

Test 1

Pressed CTRL-C and then CTRL-V

Expected result

Copies "computer-science@bmk" to input box

Actual result

As expected

## Objective: 3.b.i

Test 1

Pressed c

Expected result

Matched list as shown below, without everything after the //, which are comments which give the logic behind the order.  
computer-science@bmk //Prefix match

physics@bmk //Substring match

Actual result

As expected

### Objective: 4.a.iii

Test 1

Pressed CTRL-E after opening "root@bmk" and wrote "hello" and pressed Enter

Expected result

Option renamed to hello

Actual result

As expected

### Objective: 4.a.iv

Test 1

Pressed CTRL-E after opening "root@bmk" and wrote "hello" pressed Esc and then pressed CTRL-Q

Expected result

Nothing changes

Actual result

As expected

### Objective: 4.a.v

Test 1

Pressed CTRL-E after opening "root@bmk" and wrote "l" pressed Tab

Expected result

Input text set to "lots@lst"

Actual result

As expected

### Objective: 4.b.i

Test 1

Pressed CTRL-E after opening "root@bmk" and pressed down

Expected result

Input text set to "physics@bmk"

Actual result

As expected

## Objective: 4.b.ii

Test 1

Pressed CTRL-E after opening "root@bmk" and pressed a

Expected result

Input text set to "computer-science@bmka"

Actual result

As expected

## Objective: 4.b.iii

Test 1

Pressed CTRL-E after opening "root@bmk"

Expected result

Input text set to "computer-science@bmk"

Actual result

As expected

## Objective: 5.a.iii

Test 1

Pressed CTRL-K after opening "root@bmk" and wrote "hello" and pressed "Enter"

Expected result

"hello" added to front of option list

Actual result

As expected

## Objective: 5.a.iv

Test 1

Pressed CTRL-K after opening "root@bmk" and wrote "hello" and pressed "Esc"

Expected result

Option list doesn't change

Actual result

As expected



## Objective: 5.b.i

### Test 1

Pressed CTRL-K after opening "root@bmk"

Expected result

Input text set to ""

Actual result

As expected

## Objective: 6.a.i - iii

### Test 1

Enter "root@bmk", press CTRL-R. Then write "hello", press CTRL-A. Select physics@bmk, CTRL-E and write "world". Press Enter. Close with ESC. Open program again.

Expected result

"hello" at end of option list and "computer-science@root" removed from option list. "physics@bmk" replaced with "world"

Actual result

As expected

## Objective: 6.b.i - iii

### Test 1

Enter "data-representation-pmt-files@dir", press CTRL-R. Then write "hello", press CTRL-A. Select physics@bmk, CTRL-E and write "world". Press Enter. Close with ESC. Open program again.

Expected result

"hello" at end of option list and "2s complement" removed from option list. "Sound" replaced with "world"

Actual result

As expected

## Objective: 6.c

### Test 1

Do same thing in test for 6.a.i - iii

Expected result

"hello" at end of option list and "computer-science@root" removed from option list. "physics@bmk" replaced with "world"

Actual result

As expected

Objective: 7.a + 7.b + 7.c + 7.d

Test

I generated a file of random strings using the following script

```
#!/bin/bash
```

```
# Define the length of each string
```

```
STRING_LENGTH=3
```

```
# Define the number of lines to generate
```

```
LINES=200
```

```
# Generate random strings
```

```
for i in $(seq 1 $LINES); do
```

```
    # Generate a random string of alphanumeric characters
```

```
    RANDOM_STRING=$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w  
$STRING_LENGTH | head -n 1)
```

```
    # Output the random string to the console
```

```
    echo $RANDOM_STRING
```

```
done
```

I opened it and checked the first 5, middle 5 and the last 10 manually. The expected results matched the actual results

Objective: 8.a

Test 1

Moved configuration directory temporarily to a different location. Set the environment variable to the location I moved it to.

Expected result

Opens in same way as before I opened it

Actual result

As expected

Test 2

Moved configuration directory temporarily to a different location.

Expected result

Throws error

Actual result

As expected

## Objective: 8.b

Test 1

Removed SCRIPTS from config directory

Expected result

Throws error

Actual result

As expected

## Testing extension objectives

### Objective: 9.a

Test 1

Write hello.

Expected result

In the HISTORY file, hello is the most recent line.

Actual result

As expected

### Objective: 9.b

Test 1

Run numbers 1 to 22 in option list

Expected result

In the HISTORY file, shows 2 to 22.

Actual result

As expected

### Objective: 10

Didn't implement

### Objective: 11

Test 1

Try to compile and run on a Windows operating system

Expected result

Compiles

Actual result

Compilation error

Why test failed

I didn't end up trying to make it cross platform. Therefore it failed.

Test 2

Try to compile and run on a Linux computer

Expected result

Compiles and runs

Actual result

Expected result

Objective: 12

Didn't implement

Objective: 13.a

Test 1

Run `./program.program /path/to/home/directory` in bash

Expected result

Shows `@dir` in title after the path, showing all filenames in directory in the option list

Actual result

Expected result

Objective: 13.b

Test 2

Run `./program.program youtube.com` in bash

Expected result

Opens youtube in browser

Actual result

Expected result

## Objective: 13.c

### Test 1

Run `./program.program 'hello world'` in bash

Expected result

Searches "hello world" with search script

Actual result

Expected result

## Objective: 13.d

### Test 1

Remove `>|search` line from config directory's IDENTIFIER\_EXTENSIONS file

Run `./program.program 'hello world'` in bash

Expected result

Does nothing

Actual result

Expected result

## Objective: 14 and 15

### Test

Log prints out action of executed strings. Type in these option strings.

option string	expected action	actual action
chrome app	app	app
choose_playlist>/path/to/playlists	search	search
choose_song@rdir@rdir music>/path/to/playlists	@rdir@rdir music	@rdir@rdir music
hello world	search	search
google.com	link	link

## Objective: 16.a

### Test 1

Open revision@lst:todo

Write "hello"

Press Enter

Expected result

Adds option hello

Actual result

Expected result

Test 2

Open revision@lst:todo

Press Enter

Expected result

Selected option removed

Actual result

Expected result

## Objective: 16.b

Test 1

Open root@bmk

Write hello:flashcard>world

Expected result

Adds option with name hello

Enter toggles the data and the name so you see hello -> world or world -> hello every time you press Enter.

Actual result

Expected result

## Objective: 16.c

Test 1

Open root@bmk

Write hello:scratchpad>world

Expected result

Adds option with name hello

Enter toggles the data you are now editing the text world

Actual result

Expected result

## Objective: 17.a

Tested on video: <https://youtu.be/rEMlKu7kpp0>

## Objective: 17.b

Tested on video: [https://youtu.be/vs9\\_Lc2nuk0](https://youtu.be/vs9_Lc2nuk0)

## Objective: 17.c

Tested on video: <https://youtu.be/T71RCPMJ98c>

## Objective: 17.d

Tested on video: <https://youtu.be/nVFCQkDcgz4>

## Objective: 17.e

Tested on video: <https://youtu.be/l4qFi-B2AYl>

## Objective: 17.f

Tested on video: <https://youtu.be/mqMHtmokdpU>

## Testing evidence video

In this video, I go through the general interface of my program.

Video: <https://youtu.be/q1eX3fc6Odo>

# Evaluation

## Client feedback

After using my program for 3 days, my client gave me this feedback.

"I like its minimal interface, it makes the menu easy to navigate. The syntax is actually quite simple and easy to get used to, although difficult to understand at first. I think I enjoy the program's versatility, allowing me to quickly save or access most of my commonly used tasks. A small gripe I have with the program is the naming of option lists. I would like a way to save option lists locally tied to an option list, like in a directory, since as of now I am restricted to one name per option list. Another feature I'd like to suggest is workspaces, in which one can change the current default option list. However, I think it is overall really good and the interface feels really smooth."

## Follow up questions

I asked him some follow up questions to clarify his feedback.

**The workspace idea sounds interesting, could clarify why you think it would be useful?**  
I think it would be really helpful if, for example, I started to revise Physics and I can change my current 'workspace' to that option list.

**Are there smaller changes you would make about the interface**

I would like a feature where you can directly move options. I know you can cut them and paste them right now, but it feels a bit clunky, you know. Also, it is weird that copy

and cut “copy” different things, one doing the name/data and one the option string, maybe that would be better if it was consistent.

## How effective was the system I designed in solving my problem

Overall, having tested out my product myself for a while, I believe it has the potential to make one more productive at revision. At the start of using the program, without any scripts, it would obviously hinder my productivity, having to add new scripts to achieve desired outcomes. However, I think I got used to the system for a while, I have gotten used to it and I think it has begun to help with my day to day work. I am able to collate together a couple of links and files, and save them for the future when I want to open them again. One downside to this system has been how I decided to organise option lists. I would often want to name option lists the same thing and they could easily conflict.

## Potential improvements

After hearing my client's feedback, and having tested and used my program myself, I have identified several areas my project can improve on. Firstly, I believe that my client's feedback of adding a workspace feature is a very good idea. Being able to have change option lists connected to opening a window will greatly improve one's productivity. However, I would still like to have the default script be “root@bmk” in the startup program. As of now, I have 2 ideas of how I can implement this. The first idea is creating an initialization script that runs on computer startup, which sets workspace to “root@bmk” and adds a WORKSPACE file to the configuration directory that contains a string representing the current menu opened. My other idea is a complete revamp of the structure and workings of my program so that it is designed to run from startup. This idea is much more ambitious and tedious, but also has its advantages, such as that configuration only has to be loaded in once.

My client's other idea is to add a feature to create local option lists. I don't think this is easily implementable, however it is an idea worth considering. I believe that maybe I can try and implement a tree structure that you can store to do this.

Another idea my client mentioned to me later on was to have an easier way to open config files like scripts, perhaps by assigning them to shortcuts. I think that would be very convenient and I'd like to implement it some day.

Another way I think I can improve my product is colour coding. I could assign a colour for each type of action. This could allow users to make associations of colours to types of actions in their brain to more quickly identify what they are looking for.

Finally, I think that I could find the action of all option strings in the search menu as well as the text in the input box and make that part bold, or if it is an implicit action I can append it to the end of the option strings in a darker colour. This will remove all ambiguity of what will be executed, allowing for a better user experience.