COMP1000 Extended Referral Report

YOU ARE REQUIRED TO COMPLETE AND SUBMIT THIS ALONGSIDE YOUR CODE

The following document has been started for you. The intention is to list tests that need to be performed to evidence which requirements have been met. It is important to be honest and only claim something works if you can evidence it. Note that the assessor will be running a whole batch of tests on your code.

You should include test cases where the user enters both valid and invalid parameters.

|  |  |  |
| --- | --- | --- |
| **Name(s)** | Mina Kutenda Kamsena | **IDE**  **(VS or Qt)** |
| **Email(s)** | [Mina.kamsena@students.plymouth.ac.uk](mailto:Mina.kamsena@students.plymouth.ac.uk)  [Minakutenda@gmail.com](mailto:Minakutenda@gmail.com)  [Minakamsena@gmail.com](mailto:Minakamsena@gmail.com) |
| **GitHub URL** |  |
| Task A Complete? (Yes, No, Partially) | Yes | ? |
| Task B Complete? (Yes, No, Partially) | Partially | ? |
| For the command line tool, I deleted the build folder and hidden folder .vs | Yes |  |
| I have zipped my code and uploaded it to the DLE | Yes |  |
| I have again downloaded my code from the DLE into an empty folder, checked it is the right version, and that is all builds and runs correctly. | No |  |
| I have completed and submitted COMP1000 Extended Refferal Report Template.docx | Yes |

## Revision History

|  |  |
| --- | --- |
| 1.0 | Released to students |
| 1.01 |  |

**Part A – Command Line Tool**

To run the solution I ran my code in VS

1. Build the solution first F7
2. Open the properties window and add the arguments

(i)Lorum.txt comp1000

(ii) TaskA lorum.txt "^(\\d)\\d" -regex

(iii) TaskA lorum.txt

(iv)

1. File Reading – Open the already created text file “lorum.txt”

**Expected Result**: The program should display the contents of the file.

**Actual Result**: The program successfully reads and displays the file contents as expected.

A black and white text

Description automatically generated

2. Basic Search

1. Run the program with a search term that exists in the file: TaskA test.txt existingword
2. Run the program with a search term that doesn't exist: TaskA test.txt nonexistentword

**Expected Result**:

* For the existing word, the program should display its occurrences with line and word numbers.
* For the non-existent word, the program should not display any search results.

**Actual Result**: The program correctly identifies and displays occurrences of existing words with their line and word numbers. It also correctly handles non-existent words by not displaying any results.

A black and white text

Description automatically generated

A screenshot of a computer

Description automatically generated

3. Regular Expression Search

1. Run the program with a regex search: TaskA test.txt "^\\d+" -regex
2. Run the program with an invalid regex: TaskA test.txt "[" -regex

**Expected Result**:

* For the valid regex, the program should display matching words with their line and word numbers.
* For the invalid regex, the program should display an error message.

**Actual Result**: The program correctly handles valid regex searches, displaying matching words with their positions. It also properly catches and reports invalid regex errors.

A black and white text

Description automatically generated

4. N-gram Statistics

1. Run the program with any search term: TaskA test.txt searchterm
2. Observe the unigram and bigram statistics in the console output.

**Expected Result**: The program should display frequency counts for individual characters (unigrams) and character pairs (bigrams) in the file.**Actual Result**: The program correctly calculates and displays unigram and bigram statistics as expected.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

5. CSV Logging

1. Run the program multiple times with different search terms.
2. Check for the existence of a "results.csv" file in the same directory.
3. Open "results.csv" and examine its contents.

**Expected Result**: The "results.csv" file should exist and contain entries for each search, including filename, search term, and hit frequency.**Actual Result**: The program successfully creates and appends to the "results.csv" file, logging each search with the required information.

A number of numbers on a white background

Description automatically generated

6. Error Handling

1. Run the program without arguments: TaskA
2. Run the program with only a filename: TaskA test.txt
3. Run the program with a non-existent file: TaskA nonexistent.txt search-term

**Expected Result**:

* For insufficient arguments, the program should display an error message.
* For a non-existent file, the program should display an error message.

**Actual Result**: The program correctly handles these error cases, displaying appropriate error messages for insufficient arguments and non-existent files.

A computer screen with white text

Description automatically generated

7. Command-line Interface

1. Run the program with various combinations of arguments:
   * TaskA test.txt searchterm
   * TaskA test.txt "regex pattern" -regex
   * TaskA test.txt -regex "regex pattern"

**Expected Result**: The program should correctly interpret the command-line arguments in all cases.**Actual Result**: The program successfully interprets different argument combinations, including the optional -regex flag in various positions.

Part B – Graphical Interface

1. Using the GUI, the user must be able to open a file using a dialog box that is standard for the platform. The application will then read the contents of the file into a data structure and display it in the GUI. For full marks, the view should scroll if it does not fit.

**Expected Result**: Read the file and make sure the view should scroll if it does not fit on one page

**Actual Result**: The program successfully produces a GUI page that reveals the content of the file requested and makes sure that it can scroll if necessary

* + Attempted but failed

Appendix

https://www.w3schools.com/cpp/default.asp

https://openresearchsoftware.metajnl.com/articles/10.5334/jors.105

https://gist.github.com/gaulinmp/da5825de975ed0ea6a24186434c24fe4

https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-ngrams

https://stackoverflow.com/questions/47118015/n-gram-ignoring-line-breaks

https://www.kaggle.com/code/sauravmaheshkar/auto-completion-using-n-gram-models