User Manual

COMP 2021 Project Fall 2024

Group 8
HUANG Lingru
SHEN Linghui
TAN Rou Xin

Introduction

This project system from Group 8 is an In-Memory Virtual File System, but in the name of Comp Virtual File System (CVFS). Slightly different from the usual virtual file system, CVFS works on stimulating a file system in memory.

To start, the user is able to enter commands for the CVFS to perform instructions as requested. CVFS works on only one virtual disk at a time. Within the maximum available size of each disk, the virtual disk may contain as many files as possible, regardless of but only specify into document-typed or directory-typed files. Users are able to create new files, delete files, rename files, change the directory of the files, listing and recursively list the files.

CVFS also allows users to construct simple or composite criteria for file searching. Users are able to list all criteria and also list as well as recursively list all files according to the criteria mentioned. CVFS also allows users to save the entire virtual disk into the local path and also load the virtual disk into the system for file stimulation.

Overall, CVFS acts as an interface for users to complete file simulations either by creating a new virtual disk or loading an available virtual disk. File simulation includes searching for files from specific criteria or by file name, and the user is able to modify, create and delete files effectively.

Command

To start the System, please run the class Application.java.

You will receive the following message. Please enter your input according to the list of available commands. Case of command is not necessary to follow. Details of each command will be explained below.

```
*************************************************************************************************************************************
                              Welcome to the Command-Line Based COMP Virtual File System!
                             The system provides you with diverse commands for file management.
                           The CVFS system is completed by HUANG Ling Ru, SHEN Ling Hui, Tan Rou Xin
*************************************************************************************************************************************

Available commands: newVD, newDoc, newDir, delete, rename, changeDir, list, rList
                    newSimpleCri, criIsDoc, newNegationCri, newIsDocCri, newBinaryCri, search, rSearch
                    save, load, undo, redo, quit.
Now you can input your commands here (Case doesn't matter):
>
```

All commands below are not case-sensitive. However, all commands must be input correctly without any spelling errors to call for the correct output. Commands between the symbol “<” and “>” are guidelines for user input that you are supposed to enter according to your preference. Spaces are essential to follow during command input.

No repetition of names is allowed. Every name set by the user should be independent regardless of the object.

If commands are required to have follow-up information, inputting only the command word will not cause any issues. Instead, the system will send a prompt message to further assist

the user to input the follow-up information. Below is an example if the users forget to input the size of the disk while creating a new virtual disk.

```
Now you can input your commands here (Case doesn't matter):
> newVD
Usage: newVD <diskSize>
> newVD 1000
A new Virtual Disk is created.
```

If the system shows an error message, please retype the entire command line again with the correct follow-up information.

During file stimulation, if you receive the message "`An unexpected error occurred: Disk is full.`", it means there's no more space to save your files. Please create a new virtual disk and save your following in it or delete some files to clear up some space in the current virtual disk. An example is as below.

```
> newDoc file2 css content2
An unexpected error occurred: Disk is full
```

For any message printed "`Unexpected command. Please try again.`"
Please retype the command line with correct follow-up information as it is due to the wrong command line inputted which the system cannot understand. Examples are as below.

When a random command is received.

```
> newnewnewn 123
Unexpected command. Please try again.
```

[REQ 1]
Functionalities: create a new disk with a specific input of disk size
How can be used: `newVD <size>`

As a virtual disk is essential for CVFS to operate with the files, and only one virtual disk is allowed under the CVFS at a time. Hence, creating a new virtual disk would automatically close any previous working disk and place the current working directory as the root directory of the disk.

Below is an example of a successful creation of a new virtual disk with a size of 10000.

```
> newVD 10000
A new Virtual Disk is created.
```

[REQ 2]
Functionalities: create a new document
How can be used: `newDoc <docName> <docType> <docContent>`

Upon receiving the user command `newDoc` and all the other acceptable information, the system will create a new document in the current working directory.

Step-by-Step Instructions:
Specific details of `<docName>`, `<docType>` and `<docContent>` will be explained here.
`<docName>`: name of the document

- Limit to 10 characters
- Only alphanumeric is acceptable. (A to Z, a to z and 0 to 9)
- Compulsory to have at least one one character

`<docType>`: type of the document

- Types of docs available are “txt”, “java”, “html”, “css”. Please choose according to your usage.

`<docContent>`: content in the document (only content can be separated by space)

Below is an example of a successful creation of a new document with the name “file1”, type “css” and content “content1”.

```
> newDoc file1 css content1
Document created successfully.
> newDoc file2 txt content2
Document created successfully.
```

Troubleshooting:
Received message: `“Invalid file name”.`

- Check if the file name meets the requirements specified above. Examples of wrong file names are given below.

```
> newDoc verylongfilename css content
Invalid file name
```

```
> newDoc invalid* html content
Invalid file name
```

Received message: `“Invalid document type”`

- Check if the document type meets the requirements specified above. Examples of wrong document types is given below.

```
> newDoc file py a python file
Invalid document type. Allowed types are: txt,java,html,css
```

Received message: `“A file or directory with the same name already exists.”`

- The name created has been used in the system. Since the system cannot handle files with the same name. Please change to another name.

```
> newdoc file2 java content of file2
Document created successfully.
> newDoc file2 css content
A file or directory with the same name already exists
```

[REQ 3]
Functionalities: create a new directory
How to use: `newDIR <dirName>`

Upon receiving the user command `newDIR` and the specific name, the system will create a new directory in the current working directory.

Step-by-step instructions:
Specific details of `<dirName>` will be explained here.
`<dirName>`: name of the directory

- Limit to 10 characters
- Only alphanumeric is acceptable. (A to Z, a to z and 0 to 9)
- Compulsory to have at least one one character

Below are examples of the successful creation of the new directory.

```
> newDir Dir1
Directory created successfully.
> newDir Dir2
Directory created successfully.
```

Troubleshooting:
Received message: "`A file or directory with the same name already exists.`"

- The name created has been used in the system. Since the system cannot handle files with the same name. Please create another name.

```
> newDir Dir1
A file or directory with the same name already exists
```

Received message: "`Invalid file name`".

- Check if the file name meets the requirements specified above.

```
> newDir verylongname
Invalid file name
> newDir invalid*
Invalid file name
```

[REQ 4]
Functionalities: delete a file
How to use: `delete <fileName>`

To delete either a document-typed file or a directory-typed file created using REQ 2 or REQ 3, the command `delete <fileName>` can be used. The system will look for the specific file and remove it from the working directory.

Below is the successful deletion of the document-typed file and directory-typed file.

```
> delete file1
File deleted successfully.
> delete Dir1
File deleted successfully.
```

Troubleshooting:
Received message: `"File does not exist"`

- system could not manage to look for the file with the provided name. Please check if you have deleted the file twice or for any spelling errors

```
> delete Dir1
File does not exist
```

[REQ 5]
Functionalities: rename a file
How to use: `rename <oldFileName> <newFileName>`

To rename a file, firstly the old file name should be given and then the new file name that you want to change it to.

Step-by-step instructions:
Specific details of `<newFileName>` will be explained here.
`<newFileName>`: name of the new file name

- Limit to 10 characters
- Only alphanumeric is acceptable. (A to Z, a to z and 0 to 9)
- Compulsory to have at least one one character

Below is an example of successful renaming file name “file1” to “file3”.

```
> rename file1 file3
File renamed successfully.
```

Troubleshooting:
Received message: "`File does not exist.`"

- system could not manage to look for the file with the provided name. Please check if you have accidentally deleted the file or for any spelling errors

```
> rename file1 file4
File does not exist
```

Received message: "`Invalid new file name`"

- Check if the file name meets the requirements specified above.

```
> rename file3 verylongname
Invalid new file name
> rename file3 ---
Invalid new file name
```

Received message: "`A file or directory with the same name already exists.`"

- The name created has been used in the system. Since the system cannot handle files with the same name. Please create another name.

```
> rename file3 file2
A file or directory with the same name already exists
```

[REQ 6]
Functionalities: changing of working directory

To facilitate the changing of the working directory, please make sure if you want to change the current working directory to its parent directory.

[Changing current working to its parent directory]
How to use: `changeDir ..`

Below is a successful example.

```
> changeDir ..
Changed directory to the parent directory.
```

Troubleshooting:
Received message: "`Already in the root directory`"

- The file is already in the root directory and has no parent directory available. If you insist on changing the file to another directory, please refer to the command below.

```
> changeDir ..
Already in the root directory.
```

[Changing current working directory to another directory]
How to use: `changeDir dirName`

Below is a successful example.

```
> changeDir Dir1
Changed directory to Dir1
```

Troubleshooting:
Received message: `"Directory not found"`

- system could not manage to look for the directory with the provided name. Please check if you have accidentally deleted the directory or for any spelling errors

```
> changeDir Dir3
Directory not found
```

[REQ 7]
Functionalities: provide a list of all files found **directly** in the current working directory
How to use: `list`

All files regardless of type will be listed. The system will finalise the total number of files and total size used by the user. For each document-typed file, file name, file type and file size will also be shown. In terms of directory-typed files, file names and file sizes will be shown.

Below is a successful example.

```
> list
Files in directory /:
file3 css 56 bytes
file2 txt 56 bytes
Dir1 96 bytes
Dir2 40 bytes
Total files in the working directory: 4
Total size of the working directory: 248 bytes
```

[REQ 8]
Functionalities: provide a list of all files from the current working directory, including files in each directory
How to use: `rlist`

All files contained in the current working directory will be listed. The system will finalise the total number of files and total size used by the user. For each document-typed file, file name,

file type and file size will also be shown. In terms of directory-typed files, file names and file sizes will be shown. A slight indentation of file name is used to differentiate files in the directory.

Below is a successful example.

```
> rlist
file3 css 56 bytes
file2 txt 56 bytes
Dir1 96 bytes
  newFile java 56 bytes
Dir2 40 bytes
Total files: 4
Total size: 248 bytes
```

[REQ 9]
Functionalities: construct a simple criteria
How to use: `newSimpleCri <criName> <attrName> <op> <val>`

Upon receiving the user command `newSimpleCri` with all other acceptable information, the system will construct a new simple criterion. This criterion created can be referenced by criName. Details of the information will be explained below.

Step-by-step instructions:
Specific details of `<criName>, <attrName>, <op>, <val>` will be explained here.
`<criName>`: name of the simple criterion

- Contains exactly 2 English characters (A to Z and a to z)

`<attrname>`: type of attributes

- Types of attributes available are “name”, “type” and “size”. Please choose according to your usage.

`<op>` and `<val>` depends on `<attrName>`

- If type “name” is chosen for `<attrName>`, `<op>` should include “contains” and `<val>` should contain String with double quote (“ “).
- If type “type” is chosen for `<attrName>`, `<op>` should include “equals” and `<val>` should contain String with double quote (“ “).
- If type “size” is chosen for `<attrName>`, `<op>` can be either “**>**”, “**<**”, “**>=**”, “**<=**”, “**==**” or “**!=**”, `<val>` must be an integer.

Below are a few successful examples of different types of attributes.

```
> newSimpleCri ab size <= 56
A simple criterion is created.
> newSimpleCri cd type equals "txt"
A simple criterion is created.
> newSimpleCri ef name contains "file"
A simple criterion is created.
```

Troubleshooting:

Received message: "`Usage: newSimpleCri <criName> <attrName> <op> <val>`"

- Check if you are missing spaces between the follow-up information or miss one of the follow-up information.

```
> newSimpleCri a
Usage: newSimpleCri <criName> <attrName> <op> <val>
```

Received message: "`Criterion name must contains exactly two English letters.`"

- Check if the criterion name meets the requirements specified above.

```
> newSimpleCri abc name contains "file"
Criterion name must contains exactly two English letters.
```

Received message: "`Attribute name must be either name, type or size`"

- Check if the attribute name meets the requirements specified above.

```
> newSimpleCri ab invalid > 100
Attribute name must be either name, type or size.
```

Received message: "`Attribute is name, Op must be contained.`"

- Check if the op meets the requirements specified above when attribute is "name".

```
> newSImpleCri ab name is file
Attribute is name, Op must be contains.
```

Received message: "`Attribute is name, val must be a string in the double quote.`"

- Check if the val meets the requirements specified above when attribute is "name".

```
> newSimpleCri ab name contains file
Attribute is name, val must be a string in the double quote.
```

Received message: "`Attribute is type, Op must be equals.`"

- Check if the op meets the requirements specified above when attribute is "type".

```
> newSimpleCri cd type is txt
Attribute is type, Op must be equals.
```

Received message: "`Attributes is type, val must be a string in the double quote.`"

- Check if the val meets the requirements specified above when attribute is "type".

```
> newSImpleCri cd type equals txt
Attribute is type, val must be a string in the double quote.
```

Received message: "`Attribute is size, Op must be >, <, >=, <=, == or !=.`"

- Check if the op meets the requirements specified above when attribute is "size".

```
> newSimpleCri ef size <> 55
Attribute is size, Op must be >, <, >=, <=, ==, or !=.
```

Received message: "`Attribute is size, val must be an integer.`"

- Check if the val meets the requirements specified above when attribute is "size".

```
> newSimpleCri ef size > fivefive
Attribute is size, val must be an integer.
```

Received message: "`Criterion already exists.`"

- The criterion created has been used in the system. Since the system cannot handle same criterion. Please create one with another information.

```
> newSimpleCri ab size < 55
Criterion ab already exists.
```

[REQ 10]
Functionalities: check whether the file of simple criterion is a document
How to use: `IsDocument`

Upon receiving the user command `IsDocument`, the system creates a criterion to check whether the files contain the correct details.

Below is a successful example.

```
> newisdoccri
An IsDocument criterion is created.
```

[REQ 11]
Functionalities: construct composite criteria
How to use: `newNegation <criName1> <criName2>`
How to use: `newBinaryCri <criName1> <criName3> <logicOp> <criName4>`

Upon receiving the user command `newNegation` or `newBinaryCri`, the system will construct a composition criterion referenced by `<criName1>`. `newNegation`, named `<criName2>`, constructs the negation of the existing criterion `<criName1>`. `newBinaryCri`, construct `<criName3> <op> <criName4>`.

Step-by-step instructions:

`<criName1>`: name of the reference criterion that has to be created (could refer to REQ 9)

- Contains exactly 2 English characters (A to Z and a to z)

`<logicOp>`: logical operators

- Available operator “&&” or “||”. Please choose according to your usage.

Below are examples of the successful creation of binary and negation criteria.

```
> newNegationCri gh ab
A Negation criterion is created.
> newNegationCri ij cd
A Negation criterion is created.
> newBinaryCri kl ab && ef
```

Troubleshooting:

Received message: “`Criterion name must contains exactly two English letters.`”

- Check if the criterion name meets the requirements specified above.

```
> newNegationCri cde ab
Criterion name must contains exactly two English letters.
```

Received message: “`LogicOp is either && or ||`”

- Check if the logicOp meets the requirements specified above.

```
> newBinaryCri ef ab &| cd
LogicOp is either && or ||
```

Received message: “`Criterion <criName1> already exist.`”

- The criterion created has been used in the system. Since the system cannot handle same criterion. Please create one with another information.

```
> newNegationCri cd ab
Criterion cd already exists.

> newBinaryCri ab ab && cd
Criterion ab already exists.
```

Received message: “Criterion <criName2> doesn’t exist.”

- Composition of criteria could not happen without the creation of criterion in the first place. Please refer to REQ 9 to create a criterion before constructing composite criteria.

```
> newNegationCri cd ef
Criterion ef doesn't exist.
```

Received message: “Criterion <criName3> doesn’t exist.”
Received message: “Criterion <criName4> doesn’t exist.”

- Composition of criteria could not happen without the creation of criterion in the first place. Please refer to REQ 9 to create a criterion before constructing composite criteria.

```
> newBinaryCri ef gh && ab
Criterion gh doesn't exist.
```

[REQ 12]
Functionalities: show all defined criteria
How to use: `printAllCriteria`

Upon receiving the user command `printAllCriteria` the system will go through all defined criteria as well as their details behind. One of the details from `<attrName>`, `<op>`, `<val>`, `<logicOp>` (`&&`, `||`, `NOT`) or `<IsDocument>` will be printed.

A successful example is as below.

```
> printAllCri
ab: size <= 56
cd: type equals "txt"
ef: name contains "file"
gh: NOT size <= 56
ij: NOT type equals "txt"
kl: size <= 56 && name contains "file"
IsDocument: NOT type equals "directory"
```

Troubleshooting:
Received message: “`No criterion accessible.`”

- The system found no criterion created. Please refer to REQ 9 to create criterion.

[REQ 13]
Functionalities: search files based on criterion
How to use: `search <criName>`

Upon receiving the user command `search`, the system will go through files in the working directory and look for all files that fit the `<criName>` given to calculate for the total number of files and the total size of files that fit into `<criName>`. The system will first return number of files and size of files with a message “`Do you want to display the file that`

`satisfying the criterion? (Enter Y or N)`" Please enter "Y" to get the list of the files or "N" to continue.
Both documents and directories directly contained in curWorkDir are being searched.

Below are a few examples of searching files with different criteria and with "Y" and "N".

```
> search ab
Total number of files: 3; Total size of files: 152.
Do you want to display the file list that satisfying the criterion? (Enter Y or N
Y
file3 css 56 bytes
file2 txt 56 bytes
Dir2 40 bytes
```

```
> search cd
Total number of files: 1; Total size of files: 56.
Do you want to display the file list that satisfying the criterion? (Enter Y or N
N
OK, you can enter next command.
> search ef
Total number of files: 2; Total size of files: 112.
Do you want to display the file list that satisfying the criterion? (Enter Y or N
N
OK, you can enter next command.
> search gh
Total number of files: 1; Total size of files: 96.
Do you want to display the file list that satisfying the criterion? (Enter Y or N
Y
Dir1 96 bytes
> rsearch ij
Total number of files: 3; Total size of files: 152.
Do you want to display the file list that satisfying the criterion? (Enter Y or N
Y
file3 css 56 bytes
newFile java 56 bytes
Dir2 40 bytes
> search kl
Total number of files: 2; Total size of files: 112.
Do you want to display the file list that satisfying the criterion? (Enter Y or N
N
```

```
> search IsDocument
Total number of files: 2; Total size of files: 112.
Do you want to display the file list that satisfying the criterion? (Enter Y or N
Y
file3 css 56 bytes
file2 txt 56 bytes
```

Troubleshooting:
Received message: "`The criterion does not exist.`"

- system could not manage to look for the criterion with the provided criterion name. Please check if you have any spelling errors or refer to REQ 9 for creation of criterion.

```
> search ij
The criterion doesn't exist.
```

[REQ 14]
Functionalities: search files recursively based on a given criterion
How to use: `research <criName>`

Upon receiving the user command `rsearch`, the system will go through files in the working directory and look for all files that fit the `<criName>` given. The files saved in directory-typed files that fit the `<criName>` will also be calculated. The system calculates the total number of files and the total size of the files. The system will first show the number of files and size of files with a message "`Do you want to display the file that satisfying the criterion? (Enter Y or N)`" Please enter "Y" to get the list of the files or "N" to continue.
If the subdirectory is **empty**, we view it as a file to check, else if the subdirectory contains other files, we **look inside** to check its inner files.

Below are examples of recursively searching files with different criteria.

```
> rsearch ab
Total number of files: 4; Total size of files: 208.
Do you want to display the file list that satisfying the criterion? (Enter Y or N
Y
file3 css 56 bytes
file2 txt 56 bytes
newFile java 56 bytes
Dir2 40 bytes
```

```
> rsearch IsDocument
Total number of files: 3; Total size of files: 168.
Do you want to display the file list that satisfying the criterion? (Enter Y or N
Y
file3 css 56 bytes
file2 txt 56 bytes
newFile java 56 bytes
```

Troubleshooting:
Received message: "`The criterion doesn't exist.`"

- system could not manage to look for the criterion with the provided criterion name. Please check if you have any spelling errors or refer to REQ 9 for creation of criterion.

```
> research ij
The criterion doesn't exist.
```

[REQ 15]
Functionalities: save virtual disk into local file system
How to use: `save <path>`

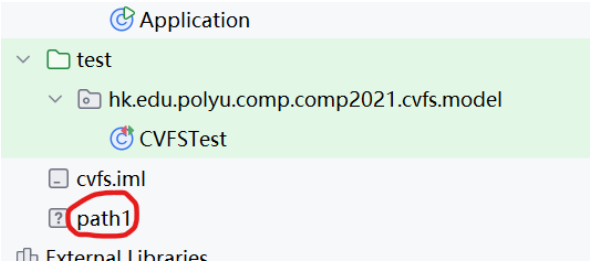
Upon receiving the user command `save`, the system saves the entire virtual disk into the same directory as the project. If the system shows "`CVFS saved successfully.`", it means the virtual disk has been exported successfully.

Step-by-step Instructions:
`<path>`: name of the virtual disk want to be shown in the local files

Below is the example of saving the virtual disk as named "path1".

```
> save path1
CVFS saved successfully.
```

Virtual disk "path1" will be saved in the same directory as the project.



Troubleshooting:
Received message: "`Error saving virtual disk.`"

- Virtual disk does not save into the local path successfully. Mainly because the file path is illegal. Please check and reread the requirements accordingly.

```
> save ?
Error saving virtual disk.
```

[REQ 16]
Functionalities: load a virtual disk from the local file into the system
How to use: `load <path>`

Upon receiving the user command `load`, the system load the entire virtual disk into the system. If the system shows "`CVFS loaded successfully.`", it means the virtual disk has been imported successfully.

Step-by-step instructions:
`<path>`: name of the virtual disk wanted to be loaded into the system

Below is a successful example.

```
> load path1
CVFS loaded successfully.
```

Troubleshooting:
Received message: "`Error loading virtual disk: File not found.`"

- Path name could not be found by the system. Please check for any spelling errors or reread the requirement accordingly.

[REQ 17]
Functionalities: stop and quit the system
How to use: `quit`

Upon receiving the user command `quit`, the system stops working and terminates from the system.

Below is a successful example.

```
> quit
Exiting COMP Virtual File System...
```

[Bouns 1]
Functionalities: saving and loading the defined search criteria when a virtual disk is saved to/loaded from the local file system.
How to use: same as save and load.

We save before quitting the program, and all the files and criteria are saved in path.

```
> save path1
CVFS saved successfully.
> quit
Exiting COMP Virtual File System...

Process finished with exit code 0
```

Then we re-access the program by running `Application.java`, entering `load <path>`, and all the files along with criteria in this path will be loaded.

```
Now you can input your commands here (Case doesn't matter):
> load path1
CVFS loaded successfully.
> rlist
file3 css 56 bytes
file2 txt 56 bytes
Dir1 96 bytes
  newFile java 56 bytes
Dir2 40 bytes
Total files: 4
Total size: 248 bytes
> printAllCri
ab: size <= 56
cd: type equals "txt"
ef: name contains "file"
gh: NOT size <= 56
ij: NOT type equals "txt"
kl: size <= 56 && name contains "file"
IsDocument: NOT type equals "directory"
>
```

[Bouns 2]
Functionalities: undo and redo of commands
How to use: `undo` / `redo`

In the case of the commands `newDoc, newDir, delete, rename, changeDir, newSimpleCri,` and `newNegation`, any wrong command inputted could be undone by inputting "`undo`" twice to return to the results of the command entered earlier. Reversely, they could also be redone by inputting "`redo`".

Below are some of the examples. Undo and Redo operations of all 8 commands are similar.

```
> newDoc doc1 txt content1
Document created successfully.
> newDir dir1
Directory created successfully.
> newSimpleCri ab size > 10
A simple criterion is created.
> newDoc doc2 txt content2
Document created successfully.
> newNegationCri cd ab
A Negation criterion is created.
> list
Files in directory /:
doc1 txt 56 bytes
dir1 40 bytes
doc2 txt 56 bytes
Total files in the working directory: 3
Total size of the working directory: 152 bytes
> printAllCri
ab: size > 10
cd: NOT size > 10
> undo
> printALlCri
ab: size > 10
```

//undo newNegation cd ab

```
> undo
> list
Files in directory /:
doc1 txt 56 bytes
dir1 40 bytes
Total files in the working directory: 2
Total size of the working directory: 96 bytes
> undo
> printALlCri
No criterion accessible.
> undo
> list
Files in directory /:
doc1 txt 56 bytes
Total files in the working directory: 1
Total size of the working directory: 56 bytes
> redo
> list
Files in directory /:
doc1 txt 56 bytes
dir1 40 bytes
Total files in the working directory: 2
Total size of the working directory: 96 bytes
> redo
> printAllCri
ab: size > 10
```

//undo newDoc doc2

//undo SimpleCri ab

//undo newDir dir1

// redo newDir dir1

//redo SimpleCri ab

Additional Resources

1. Where can I start the entire system?
   In case you missed the first part, the entire system can be run at Application.java.

2. Why some of the name created can't be acceptable?
   Majorities of the names are limited to 10 alphanumerical and must contains at least an English letter. Criterion names only accepts 2 English letters.

3. Why some errors like "...already exists" or "...doesn't exist" occurs?
   The first command (printed on the set when you enter the program) is not case-sensitive, but the file name/criterion name is (for example, "IsDocument"). Make sure you input the accurate words.

In case if you have additional enquiries towards the system that are not included in the user manual, please contact accordingly based on different parts.

- On REQ 1 to 8: lingru.huang@connect.polyu.hk
- On REQ 9 to 14: linghui.shen@connect.polyu.hk
- On REQ 15 to 17: estella.tan@connect.polyu.hk