Script for Safe File Editing

with Backup and Logging

Name: Rohini Manikandan

Submission Date: 09/04/2025

Student ID: 2434463

Anglia Ruskin University, Cambridge

|  |  |
| --- | --- |
| Table of Contents | Page No. |
| MS-DOS BATCH SCRIPT | |
| * + 1. Introduction | 2 |
| * + 1. Script Overview and Flow | 2 |
| * + 1. Code Listing with Comments | 3-6 |
| * + 1. Algorithm / Pseudocode | 7 |
| * + 1. Test Cases and Screenshots | 8-12 |
| * + 1. Conclusion | 12 |
| LINUX BASH SHELL SCRIPT | |
| * + 1. Introduction | 13 |
| * + 1. Script Overview and Flow | 13 |
| * + 1. Code Listing with Comments | 14-17 |
| * + 1. Algorithm / Pseudocode | 17-18 |
| * + 1. Test Cases and Screenshots | 19-23 |
| * + 1. Conclusion | 23 |
| BIBLIOGRAPHY | |

MS-DOS BATCH SCRIPT

# 1. Introduction

# In this report, I will explain how I have designed MS-DOS batch script for editing files. I will also explain how I have created backup for the edited files and a logfile for the backup files. It also ensures smooth execution in case of invalid inputs or file issues. The script involves both interactive as well as command-line modes. Also, the script is focused on accepting only files having extensions with .bat or .txt .

# 2. Script Overview and Flow

**In order to safely edit files, this script:-**

- Asks the user to enter a filename.  
- Checks if the file exists and confirms the filename entered.  
- Prior to opening the file for editing, makes a backup file of it if it already exists.

-Adds a timestamp to the backup action log.

-Keeps track of the last 5 backup actions by managing the log file.

**The following is the program flow:-**  
1. The script begins by determining whether command-line arguments were used to supply a filename.

2. The user is prompted to enter the filename if none is supplied.

3. The script verifies the entered file is not empty, only one argument is entered and the entered filename is having extension as either .txt or .bat

4. A new backup File is created with extension ‘.bak’ if the file exists.  
5. The new backup is used in place of any existing backups.

6. If the file does not exist, the user will be prompted to confirm whether they want to create a new one or not.

7. A “backup\_log” log file is maintained by the script, where a timestamp is appended to every backup event. If the file exceeds more than five entries in the log file, the oldest backup entry will be removed.

8. The script opens Notepad for editing the entered file.

# 3. Code Listing with Comments

@echo off

setlocal enabledelayedexpansion

**:: Display script title and separator**

echo SafeEdit Script - A tool to safely edit files with backup and logging

echo =====================================================================

**:: Set the path for the log file (same directory as this script)**

set "logfile=%~dp0backup\_log.txt"

**:: If no argument provided (interactive mode)**

if "%~1"=="" (

echo.

echo What File Do You Wish to Edit?

:ask\_filename

**:: Clear the filename variable**

set "filename="

**:: Prompt user to enter filename**

set /p "filename=Enter file name: "

**:: Trim leading/trailing spaces from input**

**:: Reference: Stack Overflow (n.d.)**

for /f "tokens=\* delims= " %%a in ("!filename!") do set "filename=%%a"

**:: If filename is empty, handle user confirmation**

if "!filename!"=="" (

echo Error: No filename entered. Please enter a valid filename.

:check

**:: Ask the user if they want to try again or exit**

set "user="

echo Do you want to Continue?

set /p user="Y/N:"

if /i "!user!"=="y" (

goto ask\_filename

) else if /i "!user!"=="n" (

echo Exiting...

pause

exit /b

) else (

echo Invalid Input

goto check

)

)

**:: Check if more than one word was entered**

for /f "tokens=2\*" %%a in ("!filename!") do (

echo Error: Too many parameters entered.

goto check

)

**:: Extract file extension and validate (.txt or .bat only)**

for %%x in ("!filename!") do set "ext=%%~xx"

if /i not "!ext!"==".txt" if /i not "!ext!"==".bat" (

echo Error: Invalid filename. Only .txt and .bat files are allowed.

goto check

)

**:: Check if the specified file exists**

if not exist "!filename!" (

echo Error: File "!filename!" does not exist.

:scroll

echo Would you like to create a new one?

set /p take\_user="Y/N:"

if /i "!take\_user!"=="y" (

echo Opening Notepad...

notepad !filename!

echo File "!filename!" was edited successfully.

goto check

) else if /i "!take\_user!"=="n" (

echo Closing creation...

goto check

) else (

echo Invalid Input!

goto scroll

)

)

**:: Call backup subroutine**

call :backup\_file "!filename!"

if "!backup\_result!" NEQ "0" (

echo Error: Backup failed.

goto check

)

**:: Log the backup and open file in Notepad**

call :log\_backup "!filename!"

start /wait notepad "!filename!"

echo File "!filename!" was edited successfully.

goto check

)

**:: Non-interactive (command-line) mode**

if not "%~2"=="" (

echo Error: Too many parameters entered.

goto check

)

**:: Assign filename from command line and clean up spaces**

set "filename=%~1"

for /f "tokens=\* delims= " %%a in ("%filename%") do set "filename=%%~nxa"

**:: Validate extension**

for %%x in ("%filename%") do set "ext=%%~xx"

if /i not "!ext!"==".txt" if /i not "!ext!"==".bat" (

echo Error: Invalid filename. Only .txt and .bat files are allowed.

goto check

)

**:: Check if file exists**

if not exist "!filename!" (

echo Error: File "!filename!" does not exist!

goto scroll

)

**:: Backup and open the file**

call :backup\_file "!filename!"

if "!backup\_result!" NEQ "0" (

echo Error: Backup failed.

goto check

)

call :log\_backup "!filename!"

start /wait notepad "!filename!"

echo File "!filename!" was edited successfully.

goto check

:backup\_file

**:: Subroutine: Creates a backup copy of the file**

**:: Input: %1 = filename**

**:: Output: Sets variable backup\_result to 0 (success) or 1 (failure)**

**:: Get full path, base name, and directory**

set "filename=%~1"

for %%f in ("%filename%") do (

set "base=%%~nf"

set "dir=%%~dpf"

)

set "backup\_file=!dir!!base!.bak"

**:: Delete existing backup if it exists**

if exist "!backup\_file!" (

del /f /q "!backup\_file!" >nul 2>&1

if exist "!backup\_file!" (

set "backup\_result=1"

exit /b

)

)

**:: Copy the file to create a backup**

copy /y "!filename!" "!backup\_file!" >nul

if errorlevel 1 (

set "backup\_result=1"

exit /b

)

**:: Indicate success**

set "backup\_result=0"

exit /b

:log\_backup

**:: Subroutine: Logs backup creation to logfile**

**:: Input: %1 = filename**

**:: Get full path of the backed-up file and timestamp**

set "backup\_file=%~1"

set "timestamp=%date% %time%"

set "bak\_file=%~n1.bak"

**:: Append backup log entry**

echo [%timestamp%] Backup created: %backup\_file% → %bak\_file% >> "%logfile%"

**:: Uses file line counting and manipulation**

**:: Reference: Microsoft Docs (n.d.), SS64 (n.d.)**

set count=0

for /f %%A in ('find /c /v "" ^< "%logfile%"') do set count=%%A

:check\_log\_size

**:: Limit log file to 5 entries**

if %count% GTR 5 (

more +1 "%logfile%" > "%logfile%.tmp"

move /y "%logfile%.tmp" "%logfile%" >nul

set count=0

for /f %%A in ('find /c /v "" ^< "%logfile%"') do set count=%%A

goto check\_log\_size

)

exit /b

# 4. Algorithm / Pseudocode

# START

# IF no command-line argument is provided

# Prompt user to enter filename

# Validate filename input (check if file is non-empty, contains no extra spaces, only one word)

# IF filename has invalid extension (.txt or .bat only)

# Display error and prompt again

# END IF

# IF file does not exist

# Ask user if they want to create a new one

# IF yes

# Open file in Notepad

# Confirm file was edited successfully

# ELSE

# Exit or return to prompt

# END IF

# END IF

# ELSE

# Use provided filename as input

# Trim extra spaces

# Validate file extension (.txt or .bat)

# IF file does not exist

# Display error and ask if user wants to create one

# Handle response accordingly

# END IF

# END IF

# IF the file exists

# Create a backup of the file with the .bak extension

# IF backup fails

# Display error and return to menu

# END IF

# Log the backup action with a timestamp in the backup log

# IF log exceeds 5 entries

# Remove the oldest log entry

# END IF

# Open the file in Notepad for editing

# Confirm file was edited successfully

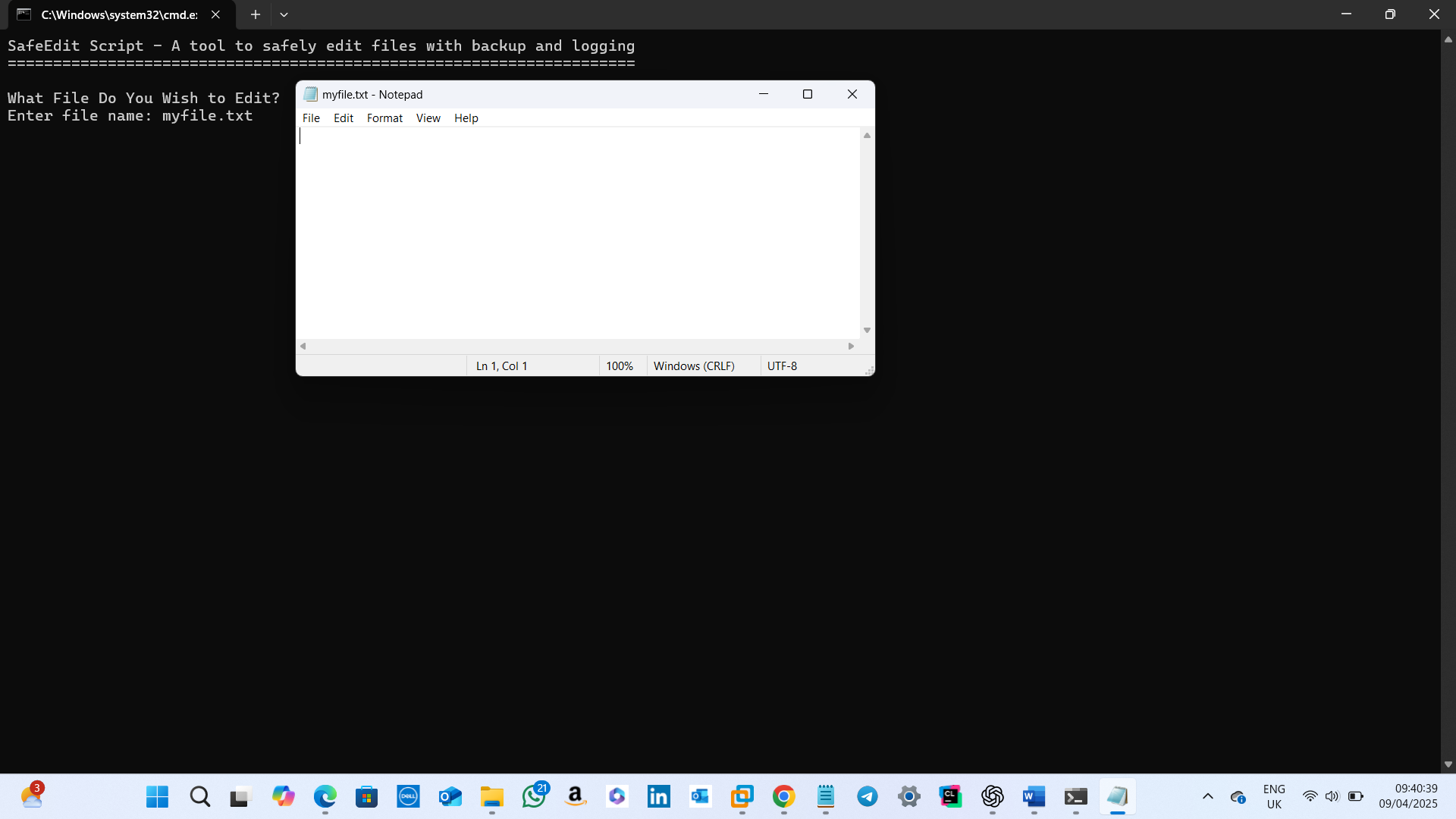
# END IF

# Limit log file to 5 entries by removing the oldest logs

# END

# 5. Test Cases and Screenshots

**Test Case 1: Valid Existent filename**

  
**Input:** A valid filename which is already existing.  
**Result:** The file is backed up, and the file is opened for editing in Notepad.

**Test Case 2: Confirming Continuation**

A screenshot of a computer

AI-generated content may be incorrect.**Input:** Asking the user if they want to continue editing after a file was edited successfully.

**Result:** If y/Y is entered , the user is asked to enter the next filename for editing.

A screenshot of a computer

AI-generated content may be incorrect.

**Input:** Asking the user if they want to continue editing after a file was edited successfully.

**Result:** If n/N is entered , prompt will be closing by displaying a closing message.

A screenshot of a computer

AI-generated content may be incorrect.

**Input:** Asking the user if they want to continue editing after a file was edited successfully.

**Result:** If anything other than y/Y/n/N is entered , an invalid input message is displayed and again the user is prompted to enter whether they want to continue or not.

**Test Case 3: Invalid filename**

A screenshot of a computer

AI-generated content may be incorrect.

**Input:** An invalid filename.  
**Result:** Displays an invalid filename message along with the file extensions allowed.

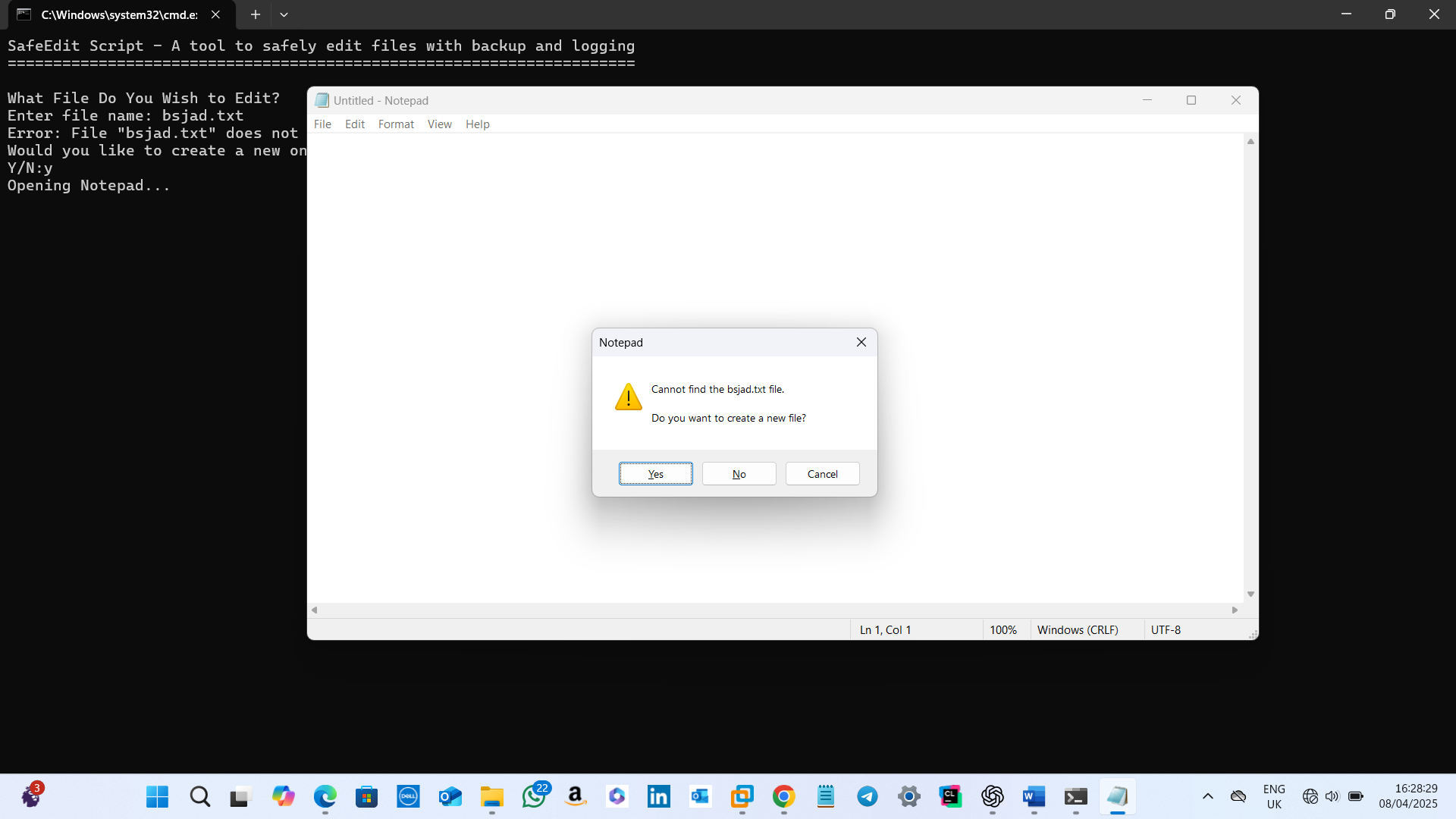
**Test Case 4: Valid Non-Existent filename**

A screenshot of a computer

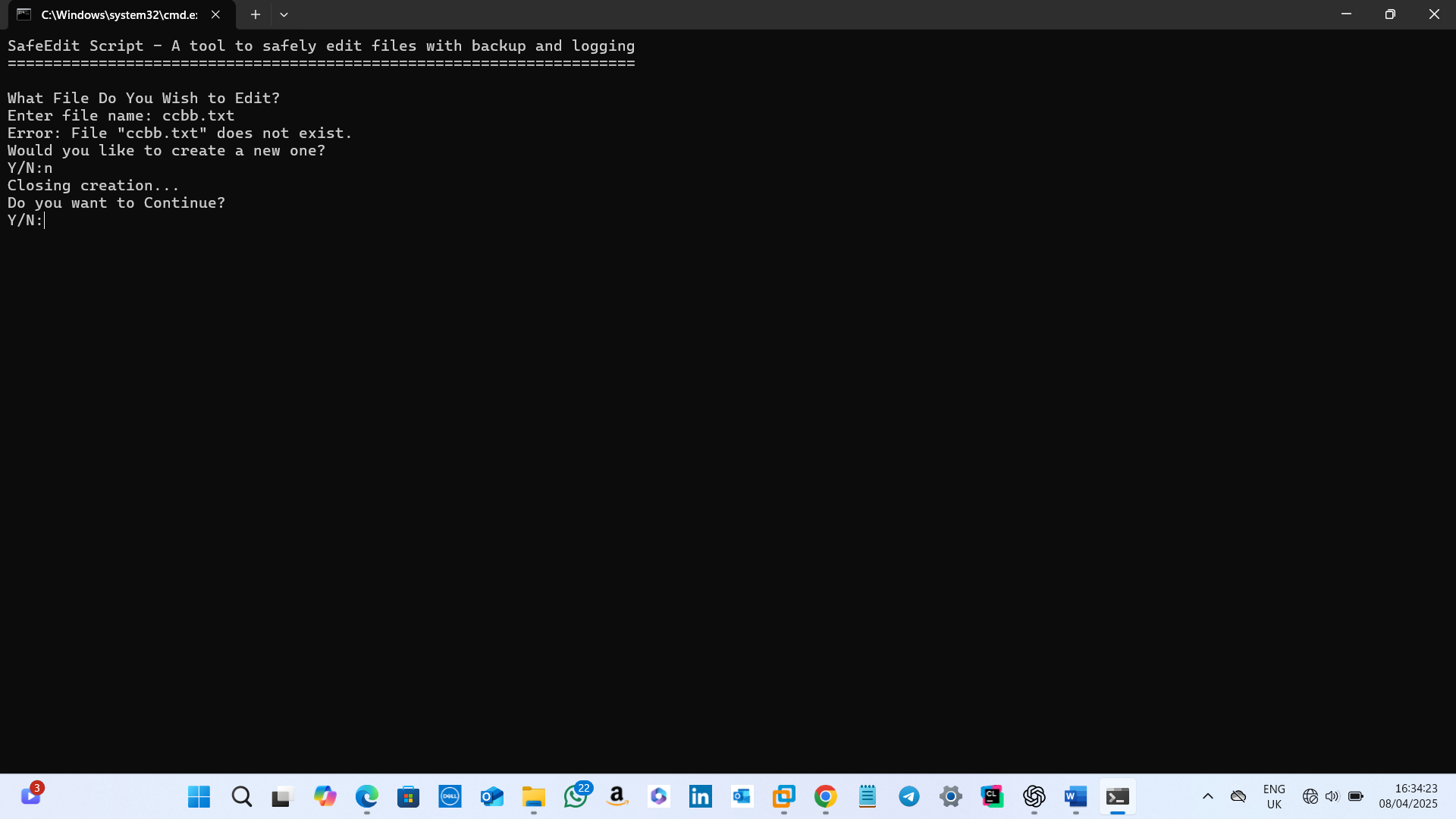
AI-generated content may be incorrect.

**Input:** A valid filename which is not existing.  
**Result:** Displays an error message and asks the user if they want to create a new one with the same filename.

**Test Case 5: Creation of a Non-Existent filename**



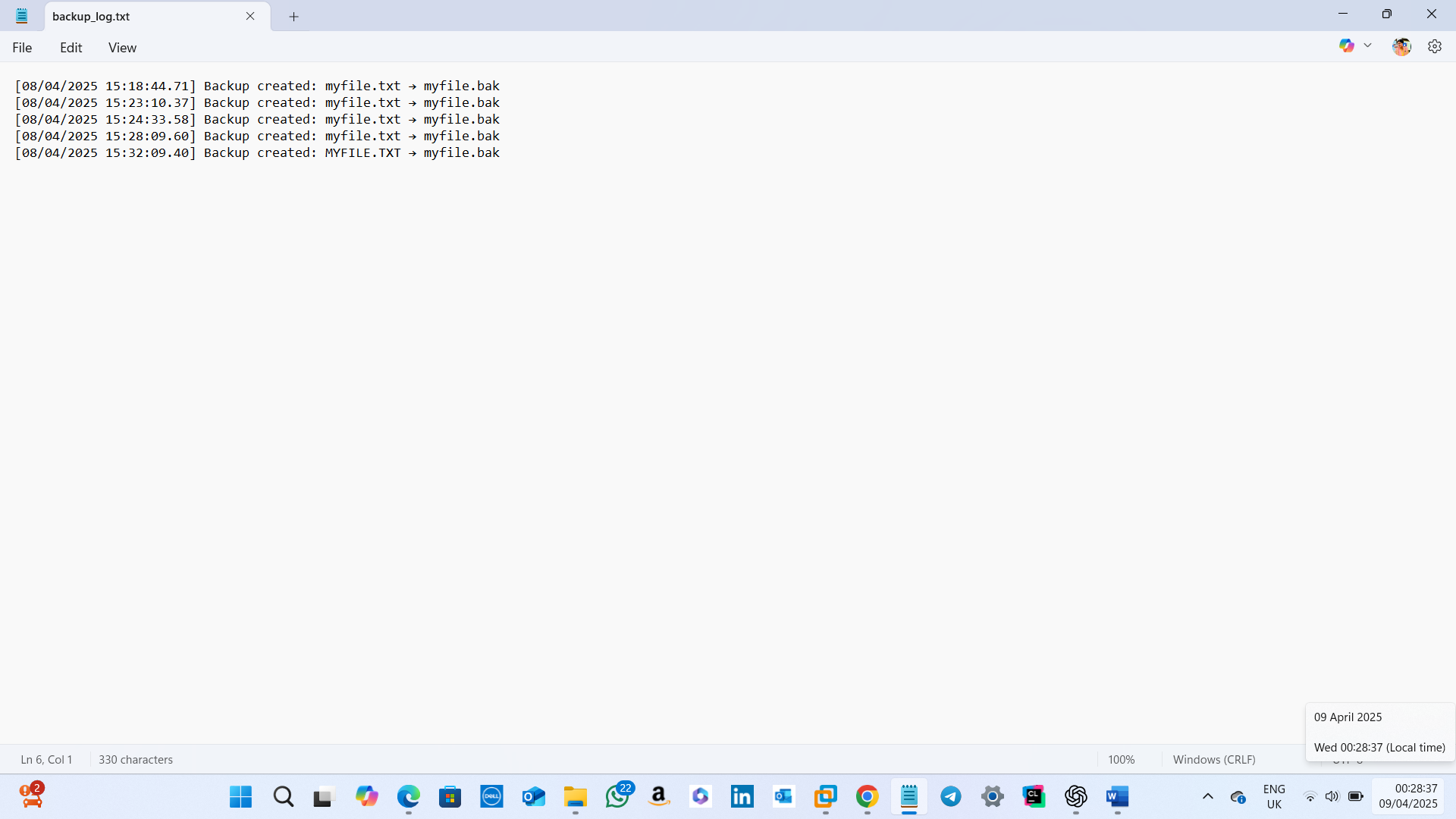
**Input:** Creation of a non-existent file.  
**Result:** If y/Y , opens notepad for creation (backup file is not created at this moment).



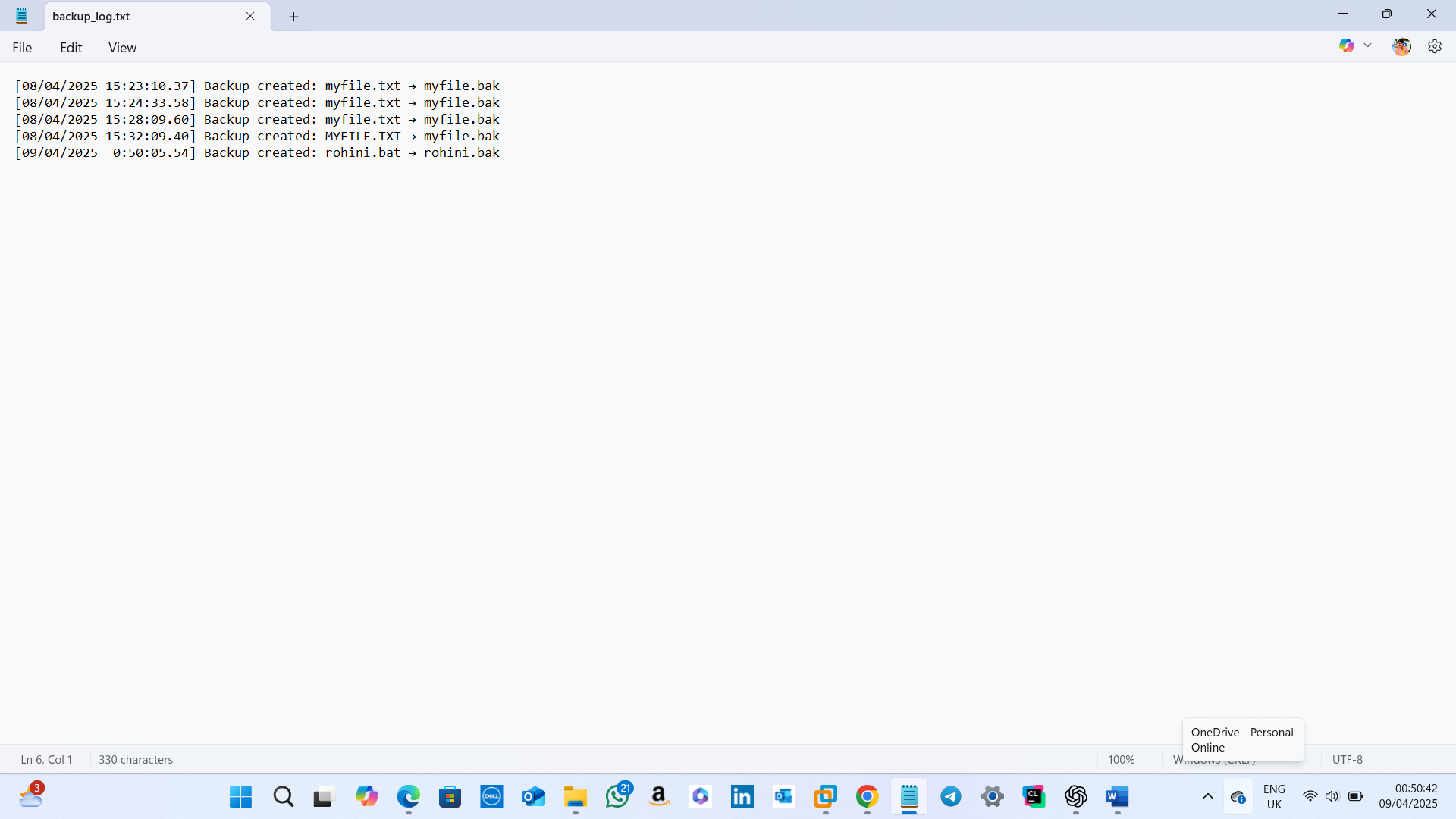
**Input:** Creation of a non-existent file.  
**Result:** If n/N , displays a closing creation message and again the user is prompted to enter whether they want to continue or not.

**Test Case 6: Multiple log entries**

This will get removed after another entry to the same file!

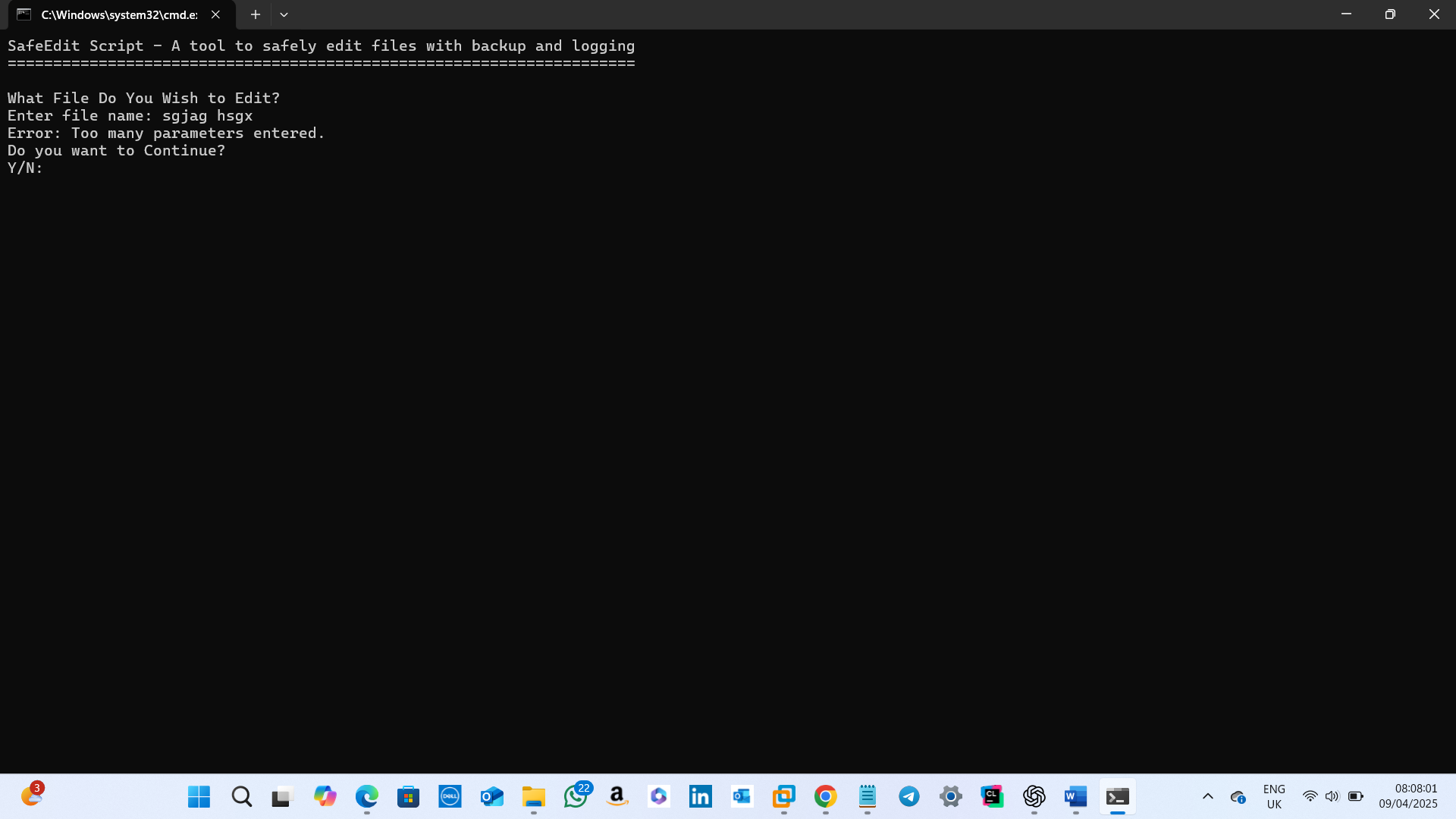


The oldest entry was removed and a latest entry was added into the backup\_log



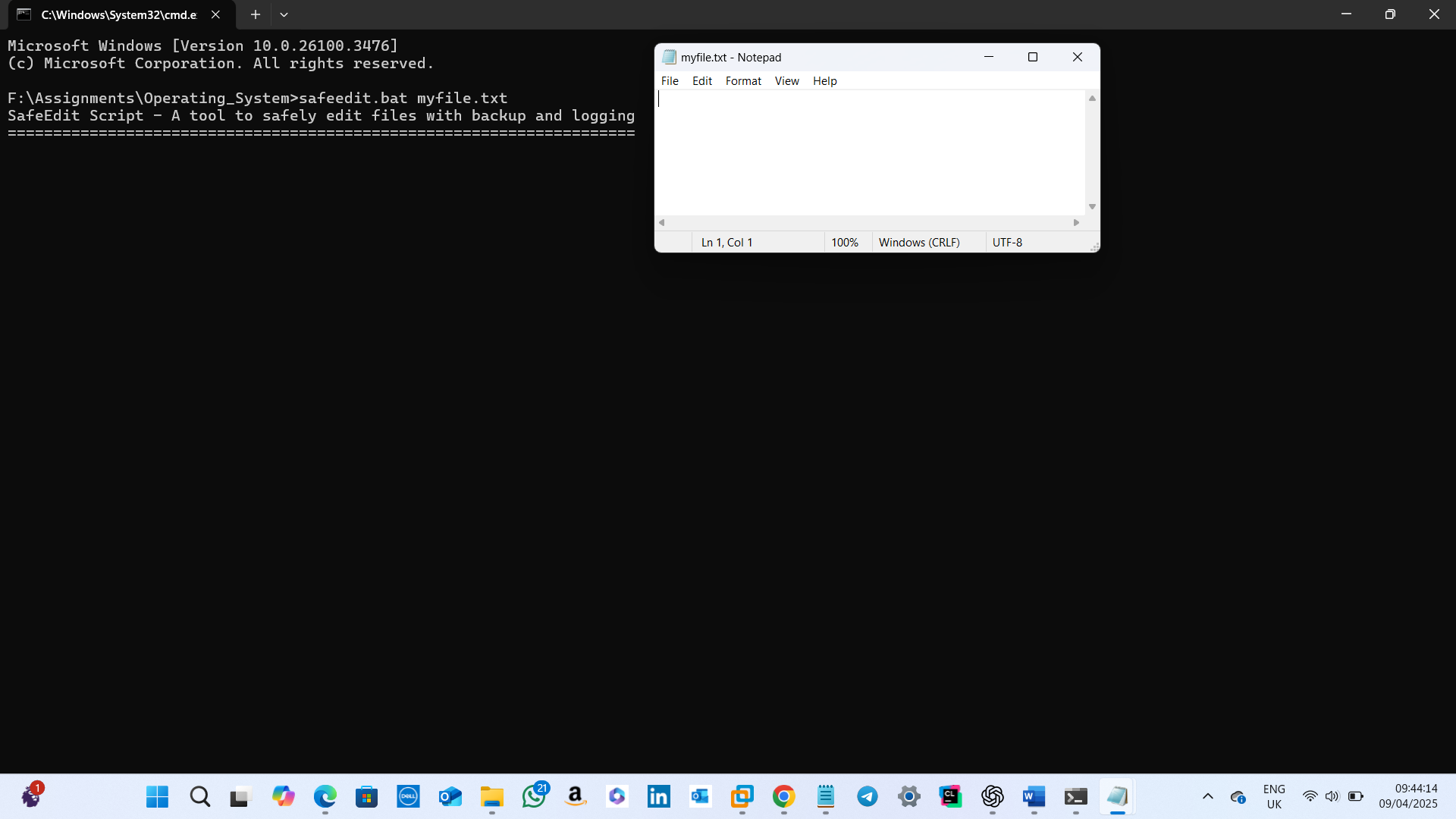
**Input:** Perform several edits to generate multiple backups.  
**Result:** Log file retains only the last 5 entries, and older entries are removed.

**Test Case 7: Check for multiple words**

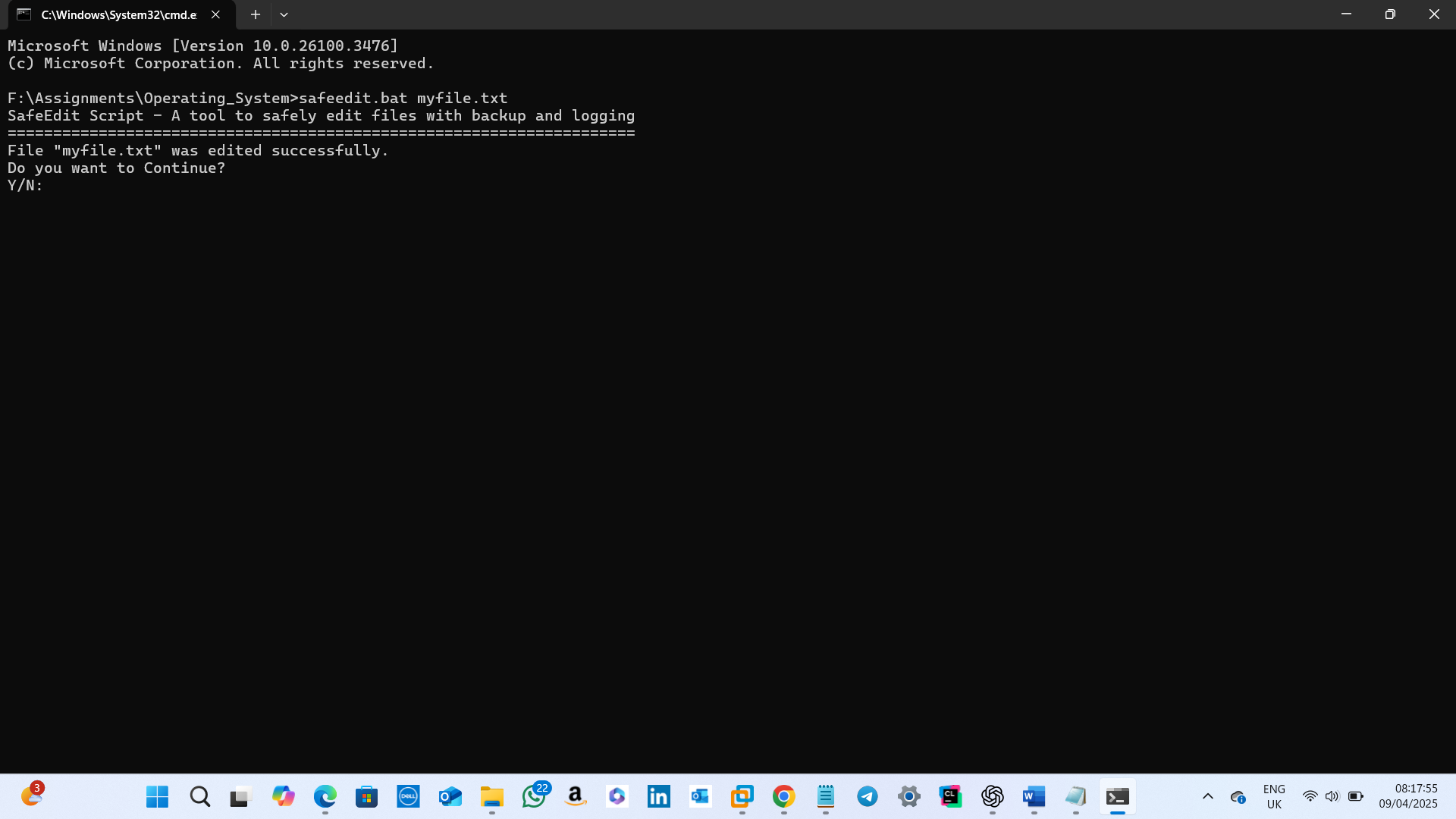


**Input:** More than one filename is entered.  
**Result:** Error message displaying too many parameters entered.

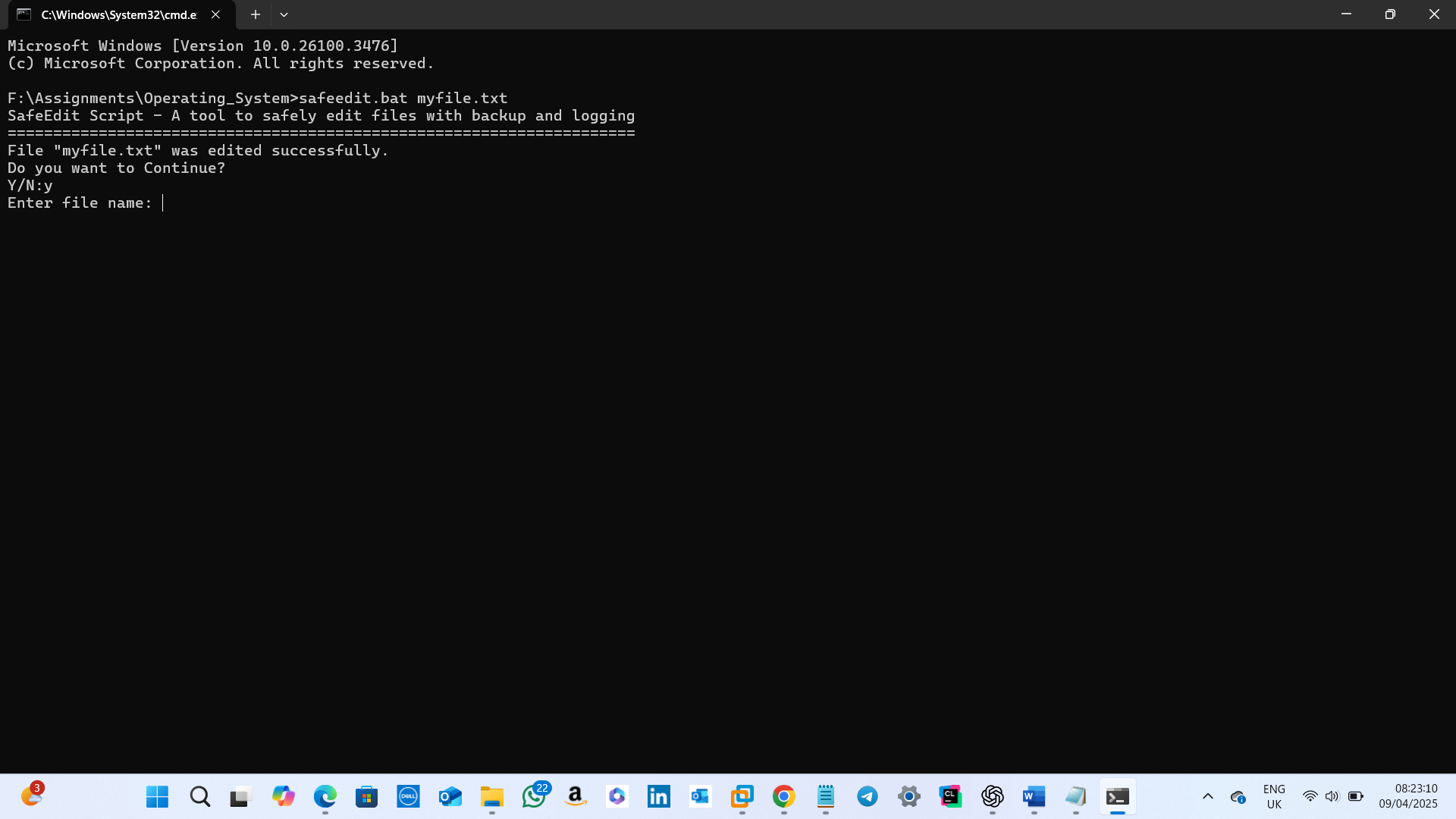
**Test Case 8: Command-line mode**



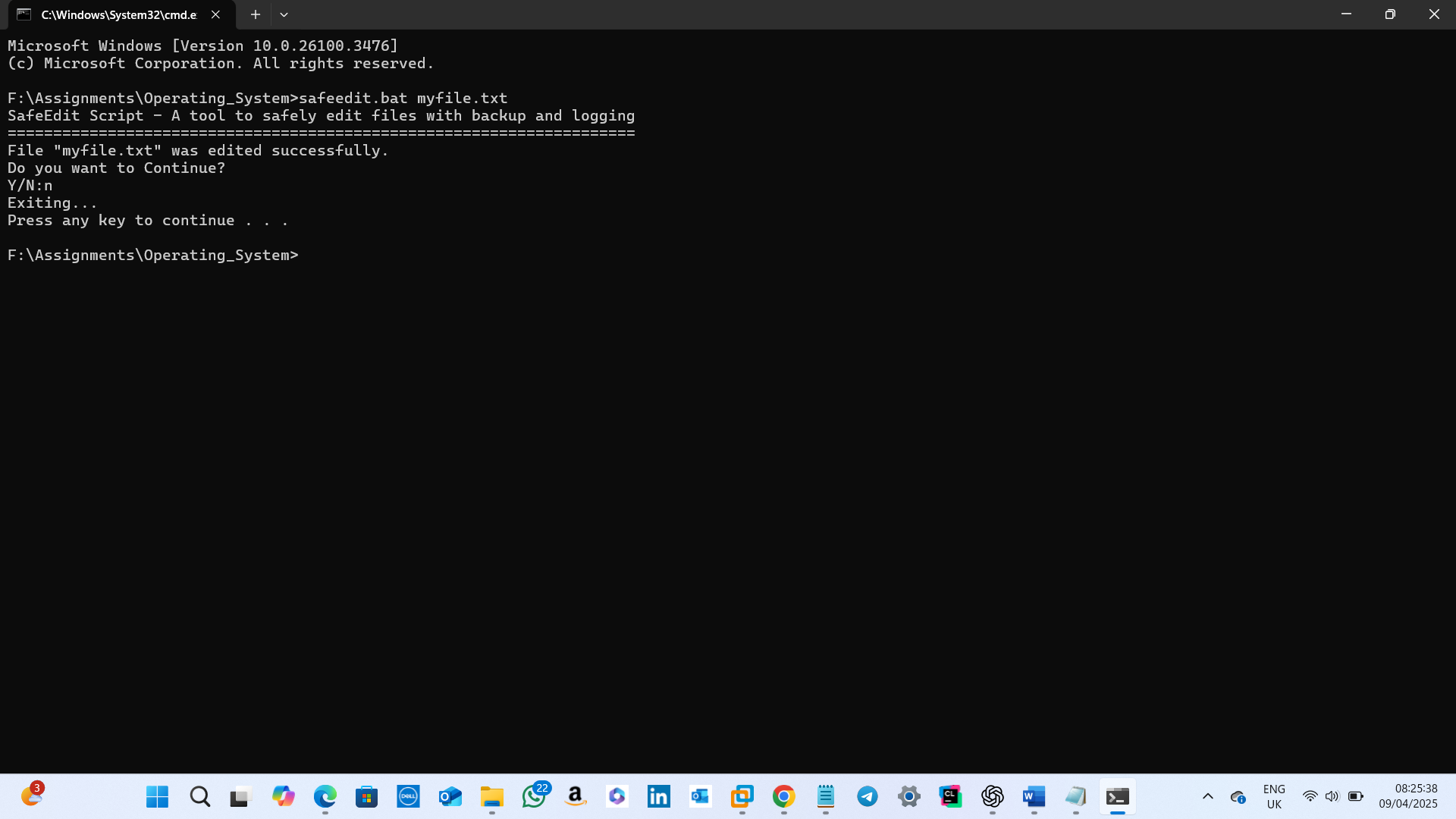
**Input:** Filename to be edited is given as argument.  
**Result:** Notepad is opened for editing.



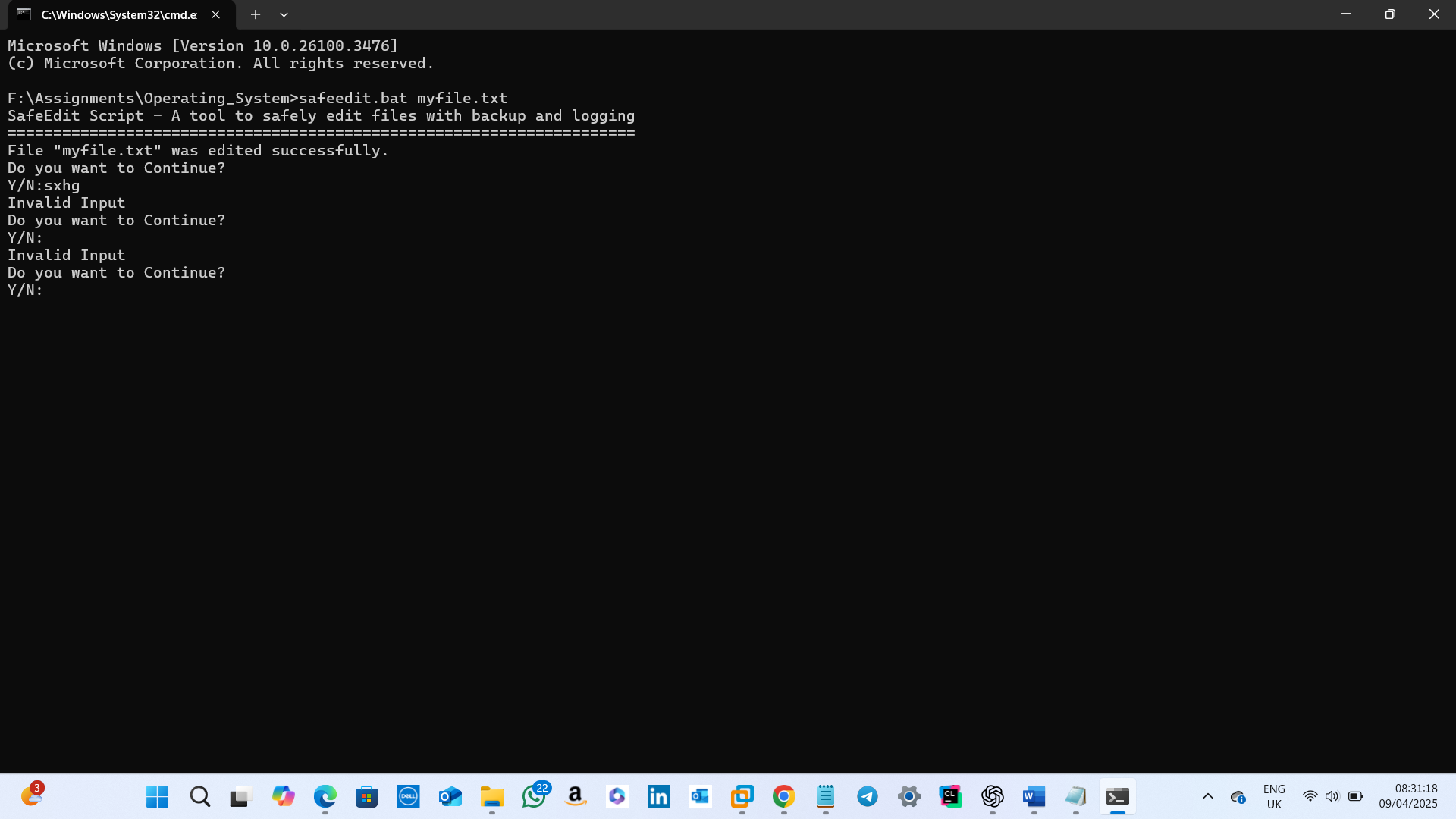
**Input:** Confirmation of continuing.  
**Result:** File edited successfully message is displayed and the user is prompted to input whether they want to continue or not(just like in interactive mode).



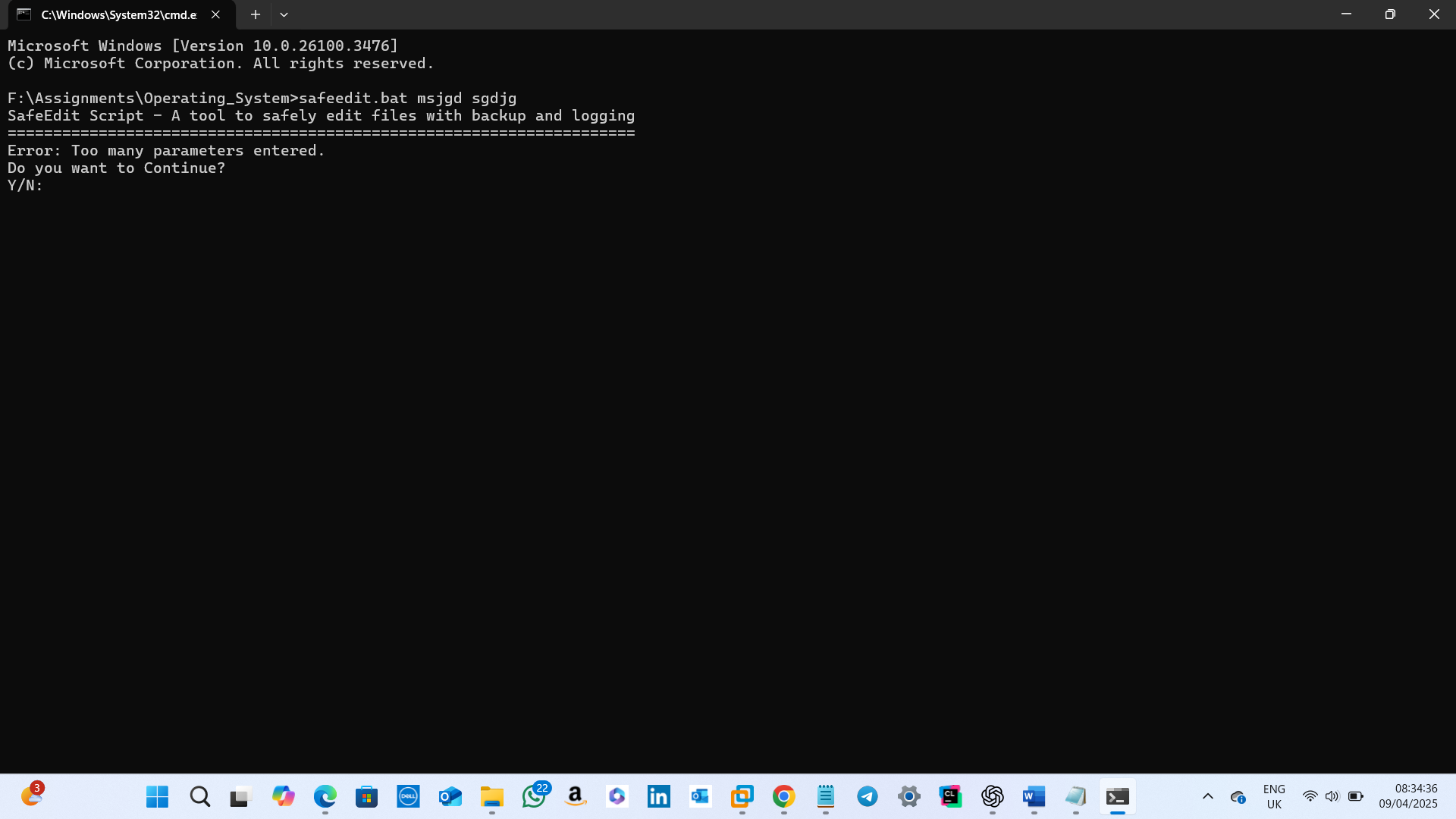
**Input:** Confirmation of continuing.  
**Result:** If y/Y is entered, the user is prompted to input the filename for editing.



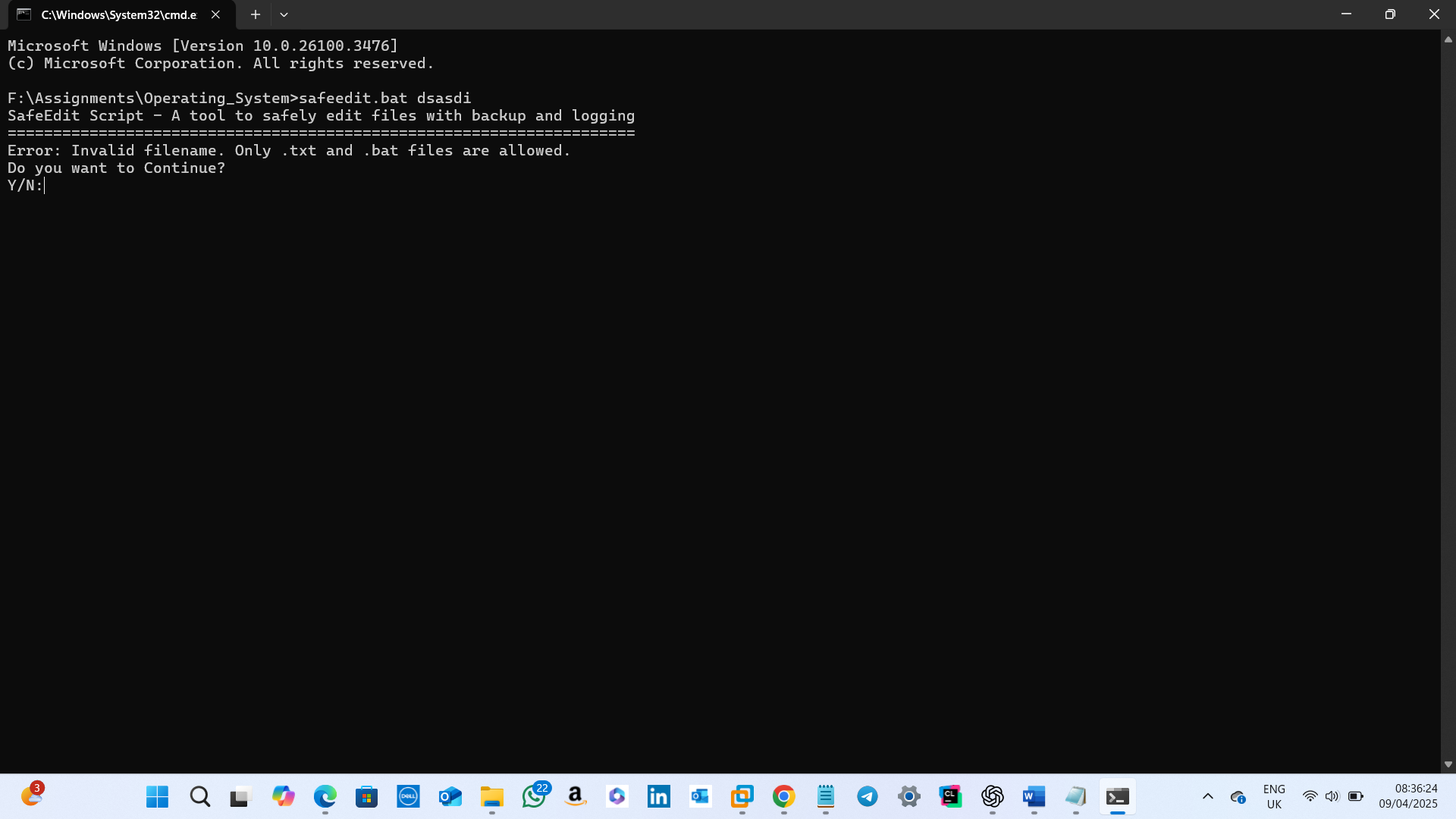
**Input:** Confirmation of continuing.  
**Result:** If n/N is entered, the user is exited from the current command line prompt and a new command line prompt will be waiting for the user to accept commands.



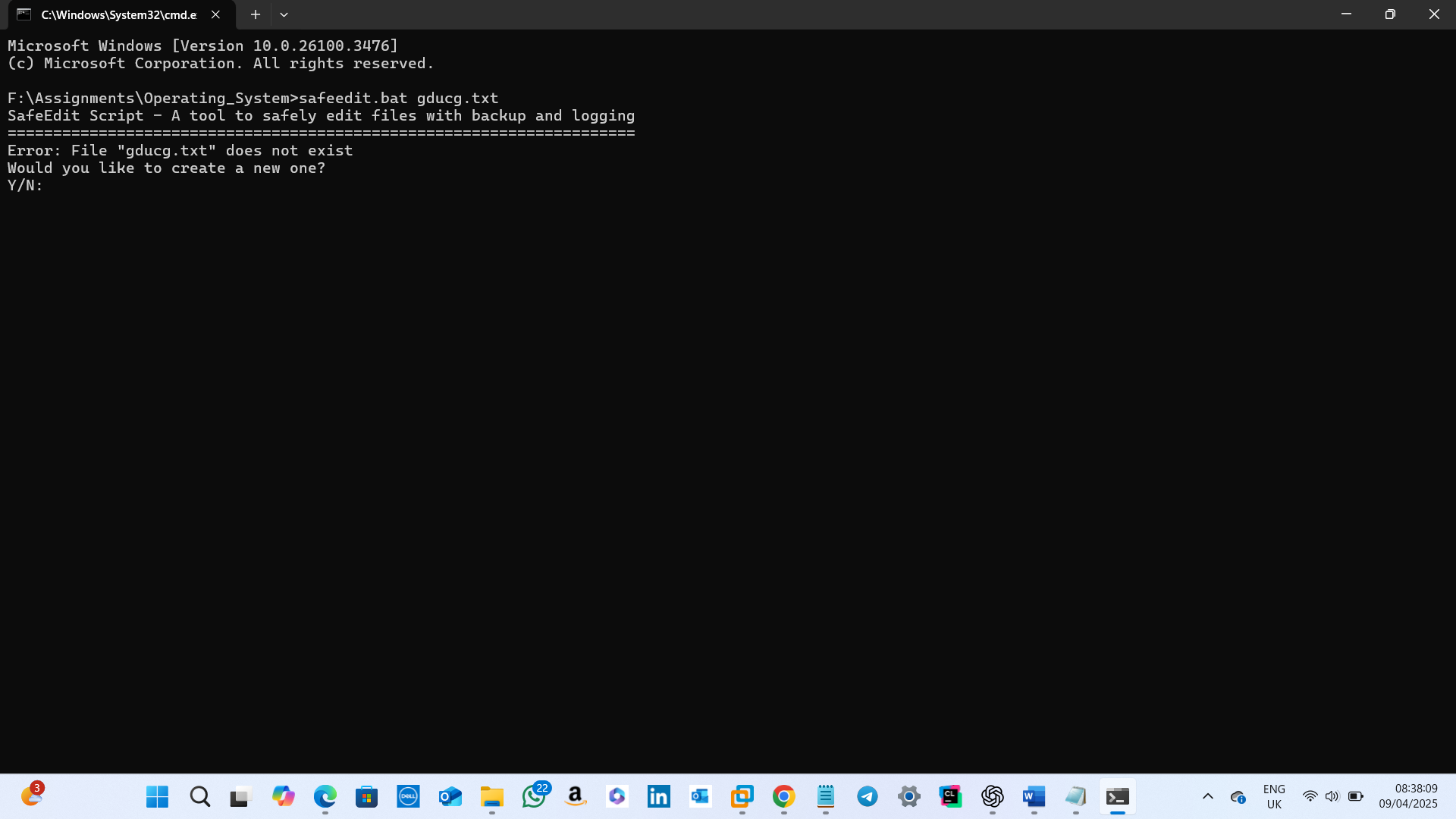
**Input:** Confirmation of continuing.  
**Result:** If anything other than y/Y/n/N is entered, an invalid output message is displayed and the user is prompted to enter whether they want to continue or not just like in interactive mode.



**Input:** More than one argument is given.  
**Result:** Error displaying too many parameters entered.



**Input:** Invalid filename is entered   
**Result:** Error displaying which all file extensions are allowed.



**Input:** Valid non-existent filename is entered.  
**Result:** Error displaying the file is not existing and the user is prompted to whether they want to create a new one or not.

# 6. Conclusion

In conclusion, the SafeEdit script performs well by ensuring the edited files are safe by creating backups and logging backup actions. Thus, thereby ensuring that users can confidently edit files.

LINUX BASH SHELL SCRIPT

# Introduction

# This report is about the Bash script I have designed for editing files. The idea is same as the MS-DOS Script. The key difference is that this script accepts only files having extensions with .sh or .txt .

# Script Overview and Flow

**The Bash script works as follows:-**

-Either a filename can be entered interactively by the user or it can be accepted as a command-line argument.

-Verifies the file's existence before continuing.

-The same base name is used to create a . bak backup of the file.

-Keeps a “backup\_log” log file, which only keeps the last five backup actions.

-The script now determines whether the current log file has reached the maximum number of permitted entries ($MAX\_LOG\_ENTRIES) before adding a new log entry. If so, the oldest log is cut off by trimming using tail, and a warning message is printed.

-Opens the file for editing in the Vi editor.

-Gratefully handles edge cases, such as missing files or an excessive number of arguments.

-If the script is in interactive mode and an empty or invalid filename is entered, the user is prompted to enter whether they want to continue editing or not.

**The following is the program flow: -**

1. If any arguments were passed, the script verifies it.
2. It attempts to backup and open the specified file if only one argument is provided.
3. The user is asked to enter the file name if no argument is provided.
4. The script:

* Confirms the existence of the file.
* It asks the user whether to continue editing or not if an invalid or empty input is entered while in interactive mode.
  + Creates a .bak backup.
  + Adds a timestamp to the log file.
  + Launches the vi file.

1. An error message appears if too many arguments are supplied.
2. If the file is not existing, user’s confirmation is taken to create a new file.

# Code Listing with Comments

# #!/bin/bash

# # File where backup logs are stored

# LOG\_FILE="backup\_log.txt"

# # Maximum number of log entries to retain in the log file

# MAX\_LOG\_ENTRIES=5

# # Function to back up the file and allow safe editing

# backup\_file() {

# local filename="$1"

# # Generate backup filename by replacing extension with .bak

# local backup="${filename%.\*}.bak"

# # Check if the file exists before backing up

# if [[ -f "$filename" ]]; then

# # Create a backup copy

# cp "$filename" "$backup"

# # Trim log file to MAX\_LOG\_ENTRIES using tail

# # Source: <https://stackoverflow.com/questions/19331837/bash-script-trimming-a-file-to-last-100-lines>

# if [[ -f "$LOG\_FILE" ]] && [[ $(wc -l < "$LOG\_FILE") -ge "$MAX\_LOG\_ENTRIES" ]]; then

# # Keep only the latest (MAX\_LOG\_ENTRIES - 1) log entries

# tail -n $((MAX\_LOG\_ENTRIES - 1)) "$LOG\_FILE" > "${LOG\_FILE}.tmp"

# mv "${LOG\_FILE}.tmp" "$LOG\_FILE"

# fi

# # Append a new log entry with timestamp

# echo "[$(date '+%Y-%m-%d %H:%M:%S')] Backup created: $filename → $backup" >> "$LOG\_FILE"

# # Open the original file in the vi editor

# vi "$filename"

# return 0

# else

# # Display an error message if file doesn't exist

# echo "Error: File '$filename' does not exist. No backup created."

# return 1

# fi

# }

# # Function to ask the user if they want to continue using the script

# prompt\_continue() {

# while true; do

# read -r -p "Do you want to continue? (yes/no): " choice

# case "$choice" in

# yes|y|Y|Yes|YES)

# return 0

# ;;

# no|n|N|No|NO)

# echo "Exiting..."

# exit 0

# ;;

# \*)

# echo "Invalid input. Please enter 'yes' or 'no'."

# ;;

# esac

# done

# }

# # Function to run the script in interactive mode

# interactive\_mode() {

# echo "==============================="

# echo " Welcome to SafeEdit (Bash)"

# echo "==============================="

# echo

# echo "This script safely edits a file by creating a .bak backup and logging the operation."

# echo

# # Loop until a valid filename is provided and processed

# while true; do

# read -r -p "What File Do You Wish to Edit? : " filename

# # Validate non-empty input

# if [[ -z "$filename" ]]; then

# echo "Filename cannot be empty. Please try again."

# prompt\_continue

# continue

# fi

# # Check if the input has spaces (indicating multiple parameters)

# if [[ "$filename" =~ \ ]]; then

# echo "Error: Too many parameters entered. Please provide only one filename."

# prompt\_continue

# continue

# fi

# # Allow only files with .txt or .sh extension

# if [[ ! "$filename" =~ \.(txt|sh)$ ]]; then

# echo "Invalid filename. Only .txt or .sh files are allowed. Please try again."

# prompt\_continue

# continue

# fi

# # Check if the file exists and proceed accordingly

# if [[ -f "$filename" ]]; then

# backup\_file "$filename"

# echo "File '$filename' edited successfully."

# prompt\_continue

# else

# # Ask user whether to create the new file

# read -r -p "File '$filename' does not exist. Do you want to create it? (yes/no): " create\_choice

# case "$create\_choice" in

# yes|y|Y|Yes)

# touch "$filename"

# echo "File '$filename' created."

# vi "$filename"

# prompt\_continue

# ;;

# no|n|N|No)

# echo "Creation aborted."

# prompt\_continue

# ;;

# \*)

# echo "Invalid input. Please enter 'yes' or 'no'."

# prompt\_continue

# ;;

# esac

# fi

# done

# }

# # Main logic for handling command-line arguments

# if [[ $# -gt 1 ]]; then

# # Error if too many arguments are provided

# echo "Error: Too many parameters entered. Please provide only one filename or run without arguments for interactive mode."

# exit 1

# elif [[ $# -eq 1 ]]; then

# # One argument provided — validate and act

# filename="$1"

# if [[ "$filename" =~ \.(txt|sh)$ ]]; then

# if [[ -f "$filename" ]]; then

# backup\_file "$filename"

# else

# echo "File '$filename' does not exist."

# read -r -p "Do you want to create it? (yes/no): " create\_choice

# case "$create\_choice" in

# yes|y|Y|Yes|YES)

# touch "$filename"

# echo "File '$filename' created."

# vi "$filename"

# ;;

# no|n|N|No|NO)

# echo "Creation aborted."

# ;;

# \*)

# echo "Invalid input. Please enter 'yes' or 'no'."

# ;;

# esac

# fi

# else

# # File extension not allowed

# echo "Invalid filename. Only .txt or .sh files are allowed."

# fi

# else

# # No arguments — run the interactive prompt

# interactive\_mode

# fi

# 4. Algorithm / Pseudocode

**BEGIN**

SET LOG\_FILE to "backup\_log.txt"

SET MAX\_LOG\_ENTRIES to 5

FUNCTION backup\_file(filename)

SET backup ← filename with ".bak" replacing its extension

**IF** file named filename exists THEN

COPY filename TO backup

DISPLAY "[timestamp] Backup created: filename → backup"

**IF** LOG\_FILE exists AND number of lines in LOG\_FILE ≥ MAX\_LOG\_ENTRIES THEN

DISPLAY "Log file exceeds MAX\_LOG\_ENTRIES entries. Deleting the oldest log."

**ENDIF**

CREATE TEMP\_LOG by:

- TAIL last (MAX\_LOG\_ENTRIES - 1) lines from LOG\_FILE (if exists)

- APPEND "[timestamp] Backup created: filename → backup"

OVERWRITE LOG\_FILE with TEMP\_LOG

OPEN filename IN vi editor

RETURN success

**ELSE**

DISPLAY "Error: File 'filename' does not exist. No backup created."

RETURN failure

**ENDIF**

ENDFUNCTION

FUNCTION interactive\_mode()

DISPLAY welcome banner and explanation

LOOP

PROMPT "What File Do You Wish to Edit?" AND READ filename

**IF** filename is NOT empty THEN

CALL backup\_file(filename)

**IF** backup\_file is successful THEN

BREAK LOOP

**ENDIF**

**ELSE**

DISPLAY "Filename cannot be empty. Please try again."

**ENDIF**

ENDLOOP

ENDFUNCTION

MAIN

**IF** number of arguments > 1 THEN

DISPLAY "Error: Too many parameters entered."

EXIT with error

**ELSE IF** number of arguments = 1 THEN

CALL backup\_file(argument)

**ELSE**

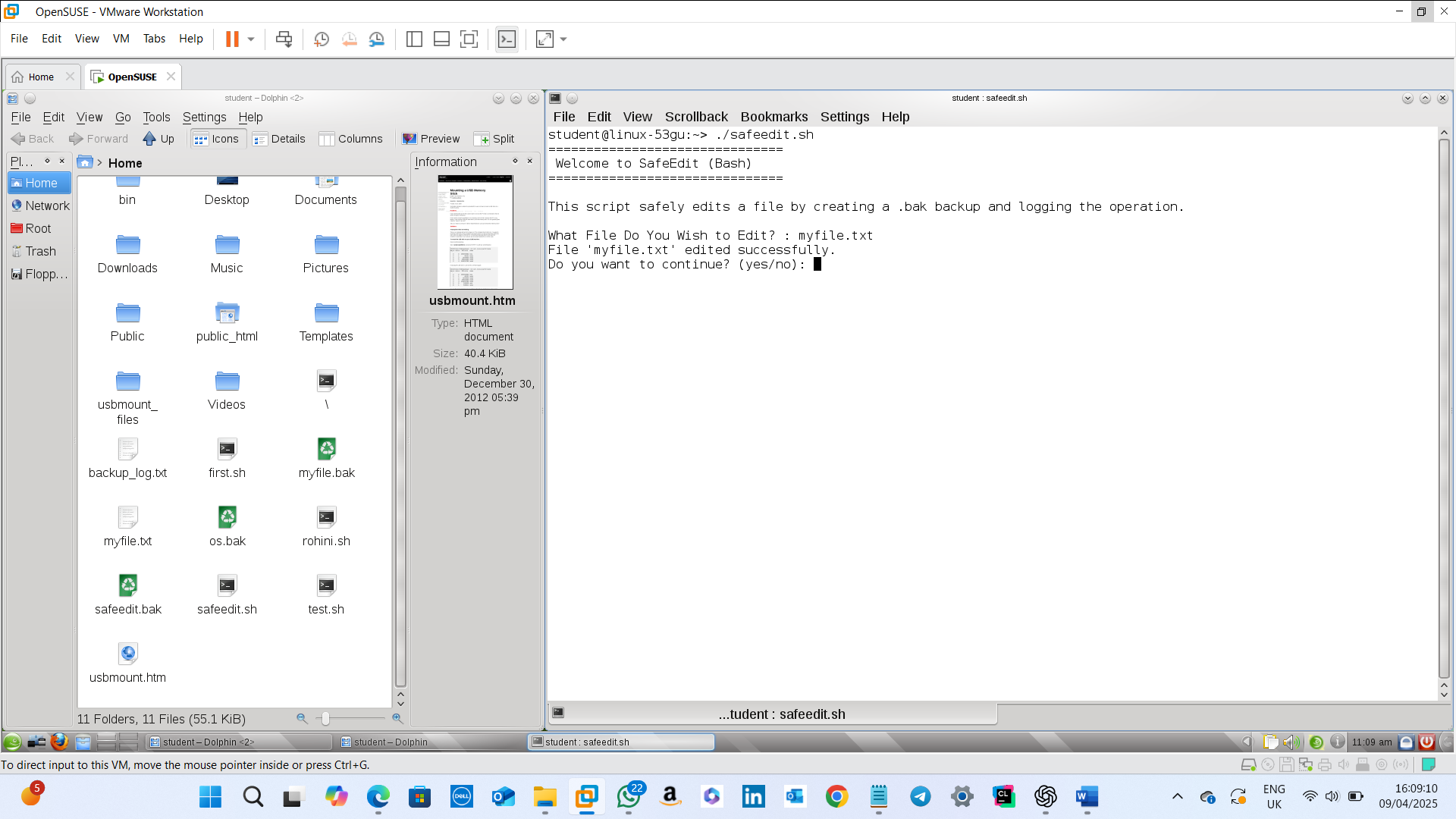
CALL interactive\_mode()

**ENDIF**

**END**

**5. Test Cases and Screenshots**

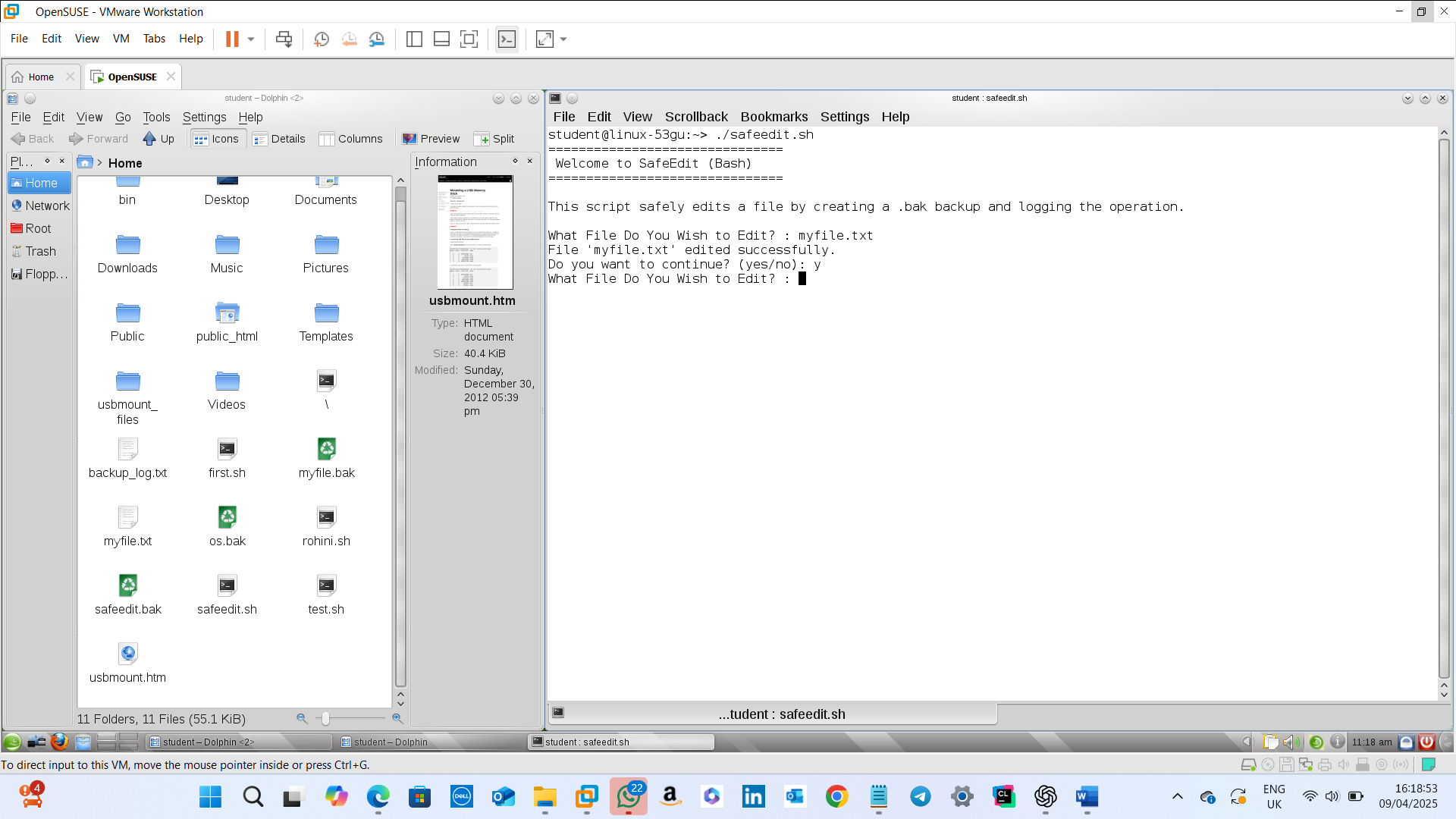
**Test Case 1: Valid file provided**



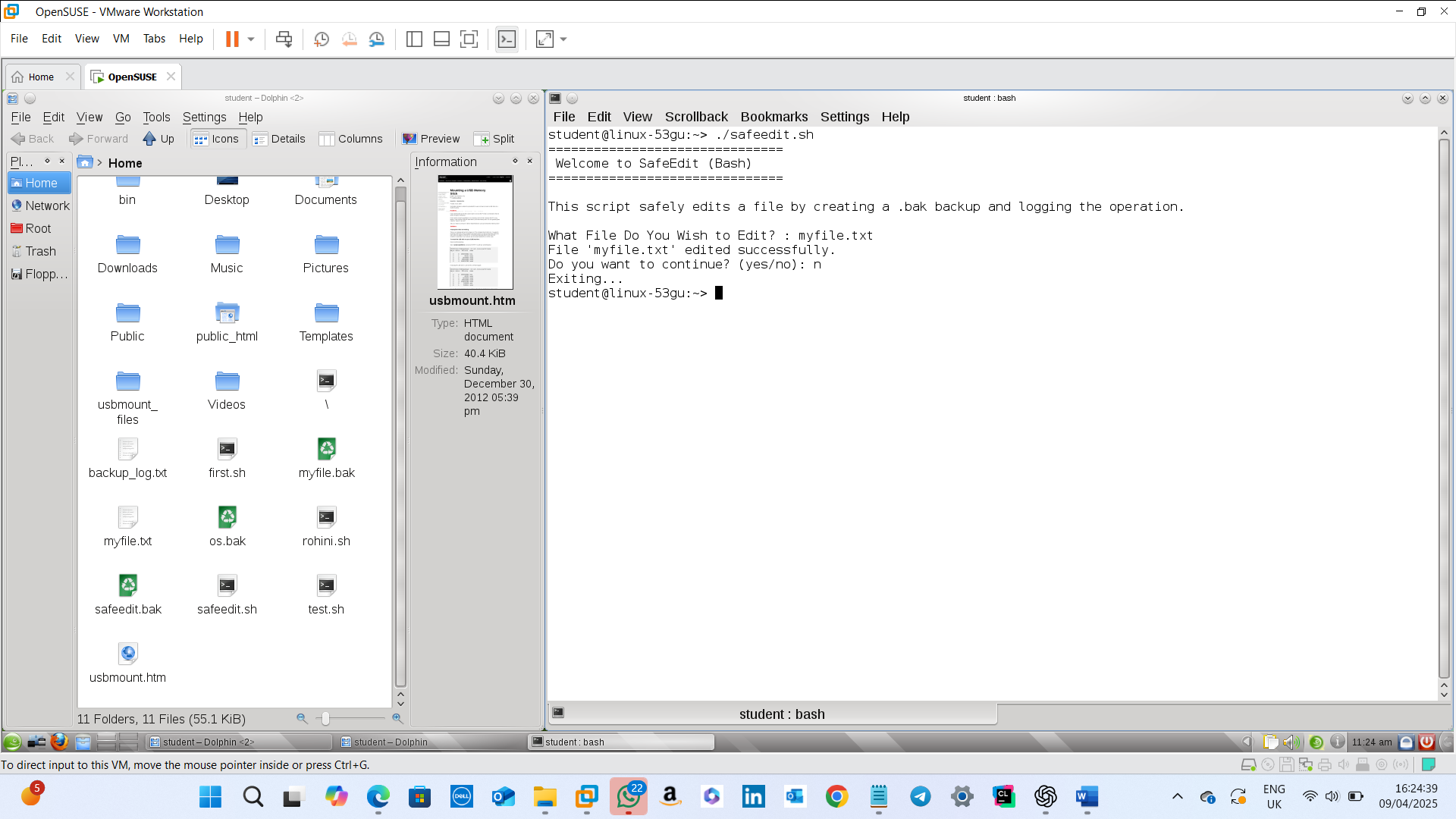
**Input:** Valid existing filename is entered.

**Result:** Opens Vi for editing and after saving that using :wq , file edited successfully message is displayed.

**Test Case 2: Confirming Continuation**

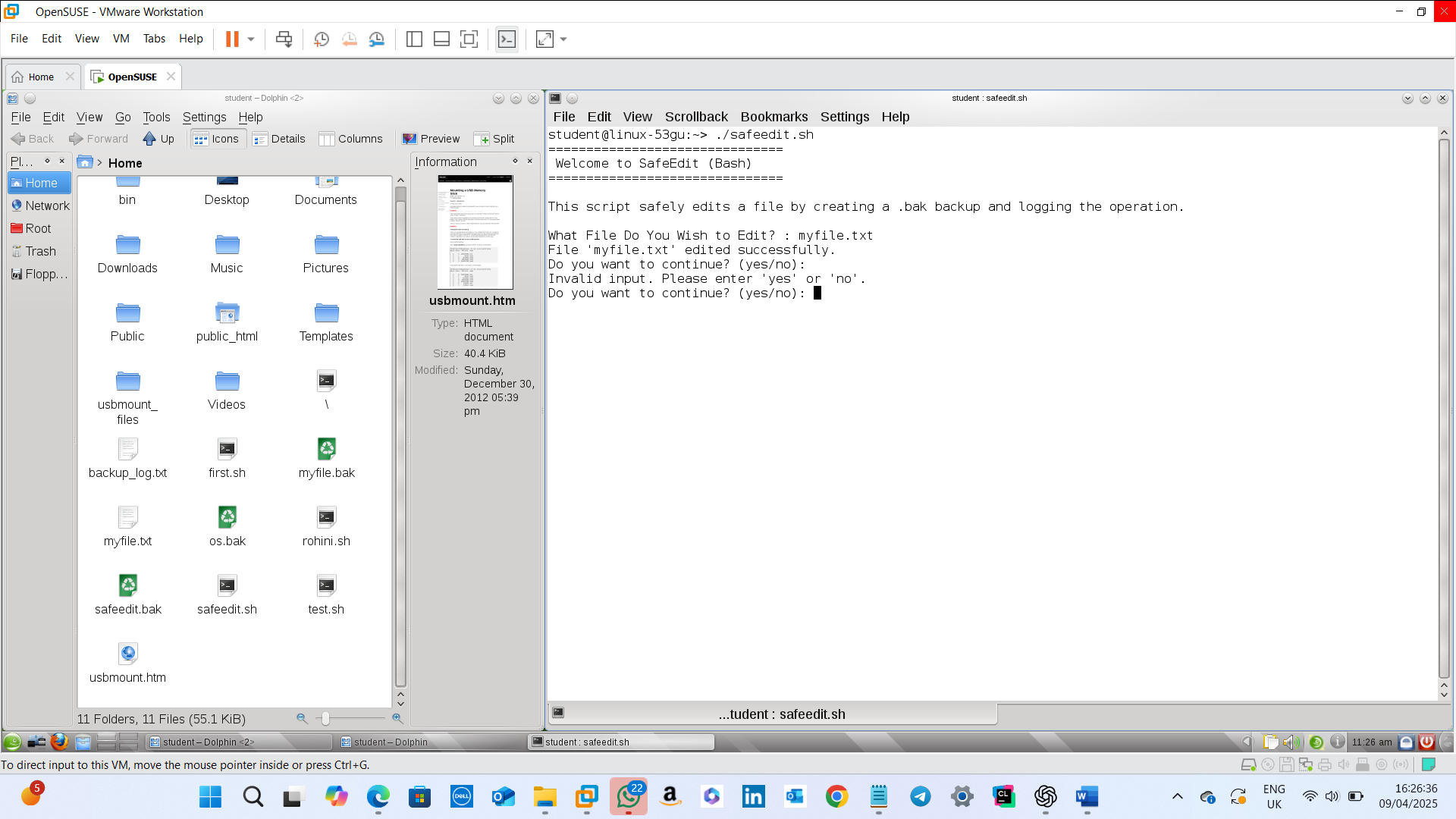
**Input:** The user is asked to enter whether they want to continue or not.

**Result:** If yes, the user is prompted to enter a filename for editing.



**Input:** The user is asked to enter whether they want to continue or not.

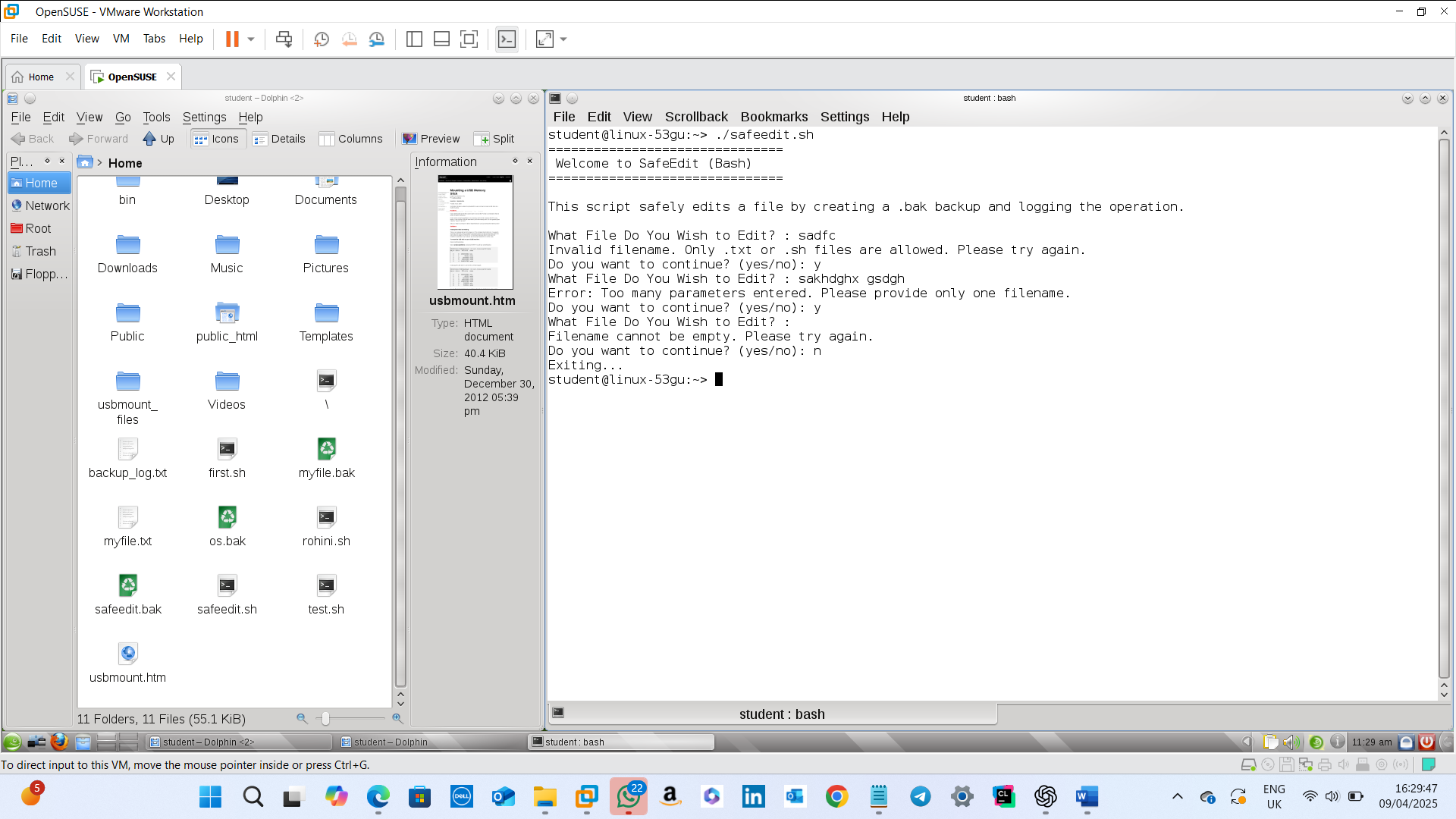
**Result:** If no, the user is exited from the prompt.



**Input:** The user is asked to enter whether they want to continue or not.

**Result:** If anything other than yes/no is entered, an invalid input message is displayed and the user is required to enter a valid entry there.

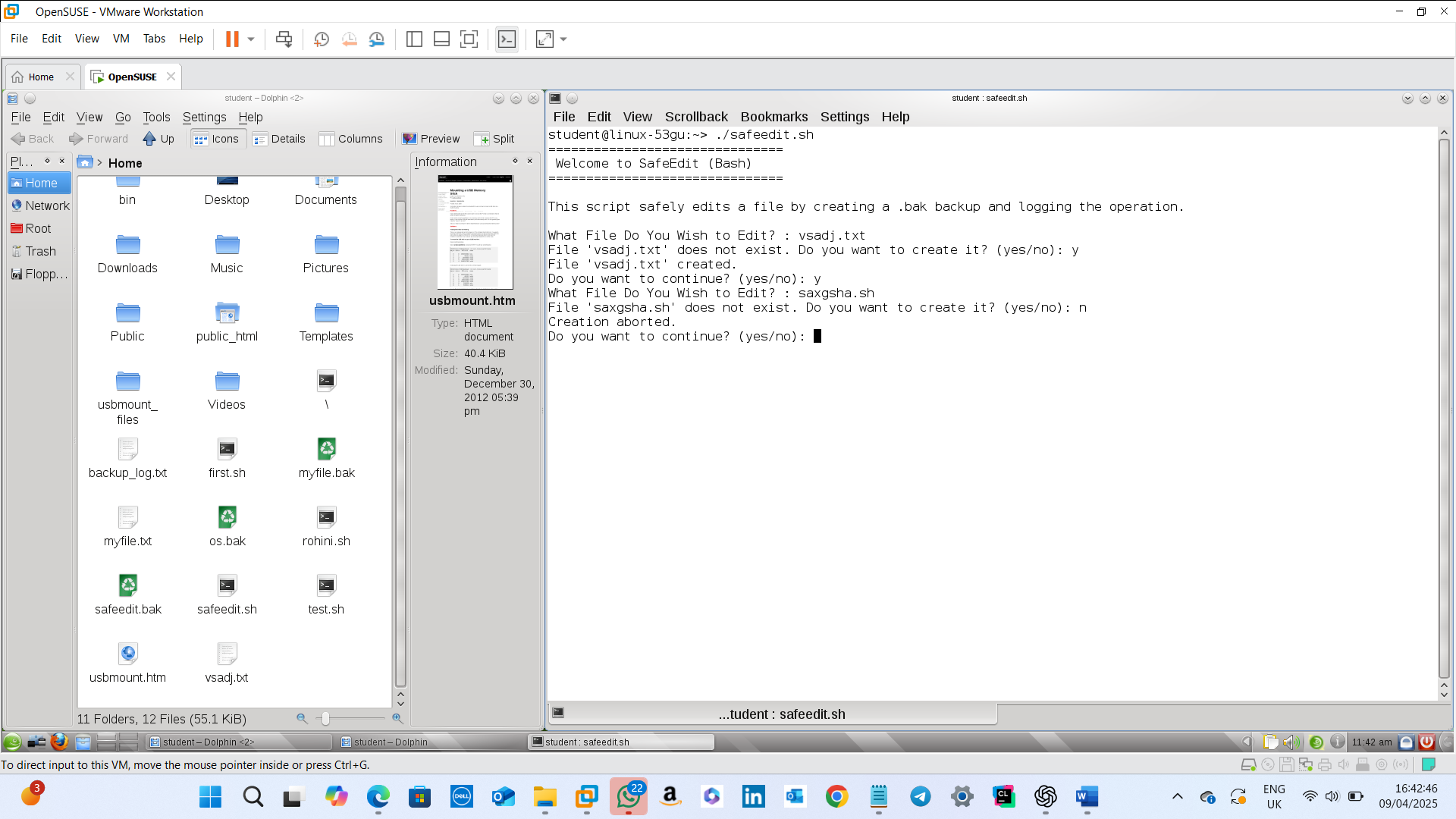
**Test Case 3: Invalid Filenames**



**Input:** Invalid inputs like empty filename, invalid filename and more than one parameter is typed.

**Result:** If the filename is invalid, allowed extension files are displayed; If the filename is empty, “filename cannot be empty” message is displayed; If more than one parameter is given, the user is asked to enter only one parameter.

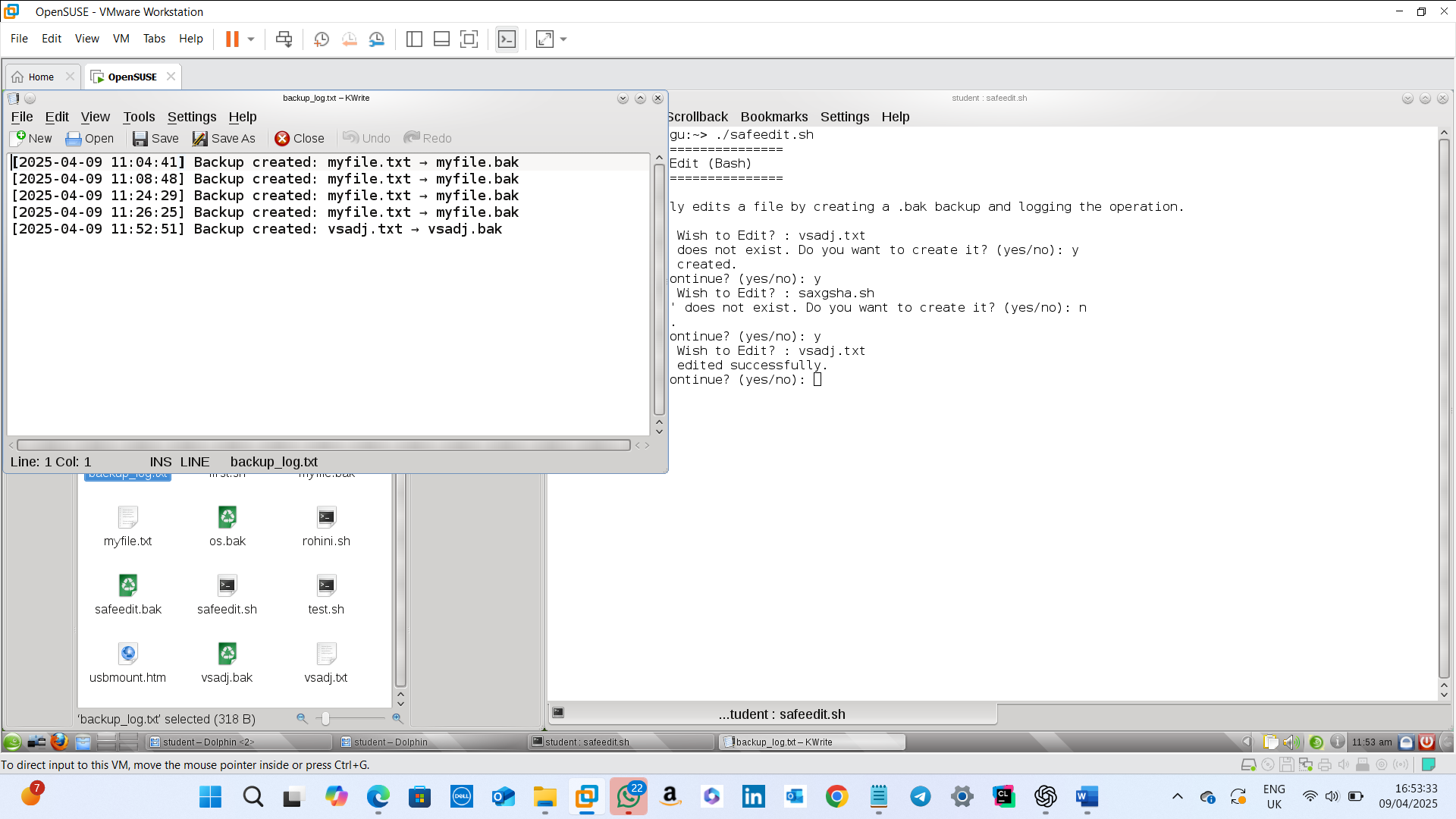
**Test Case 4: Valid Non-Existent Filename**



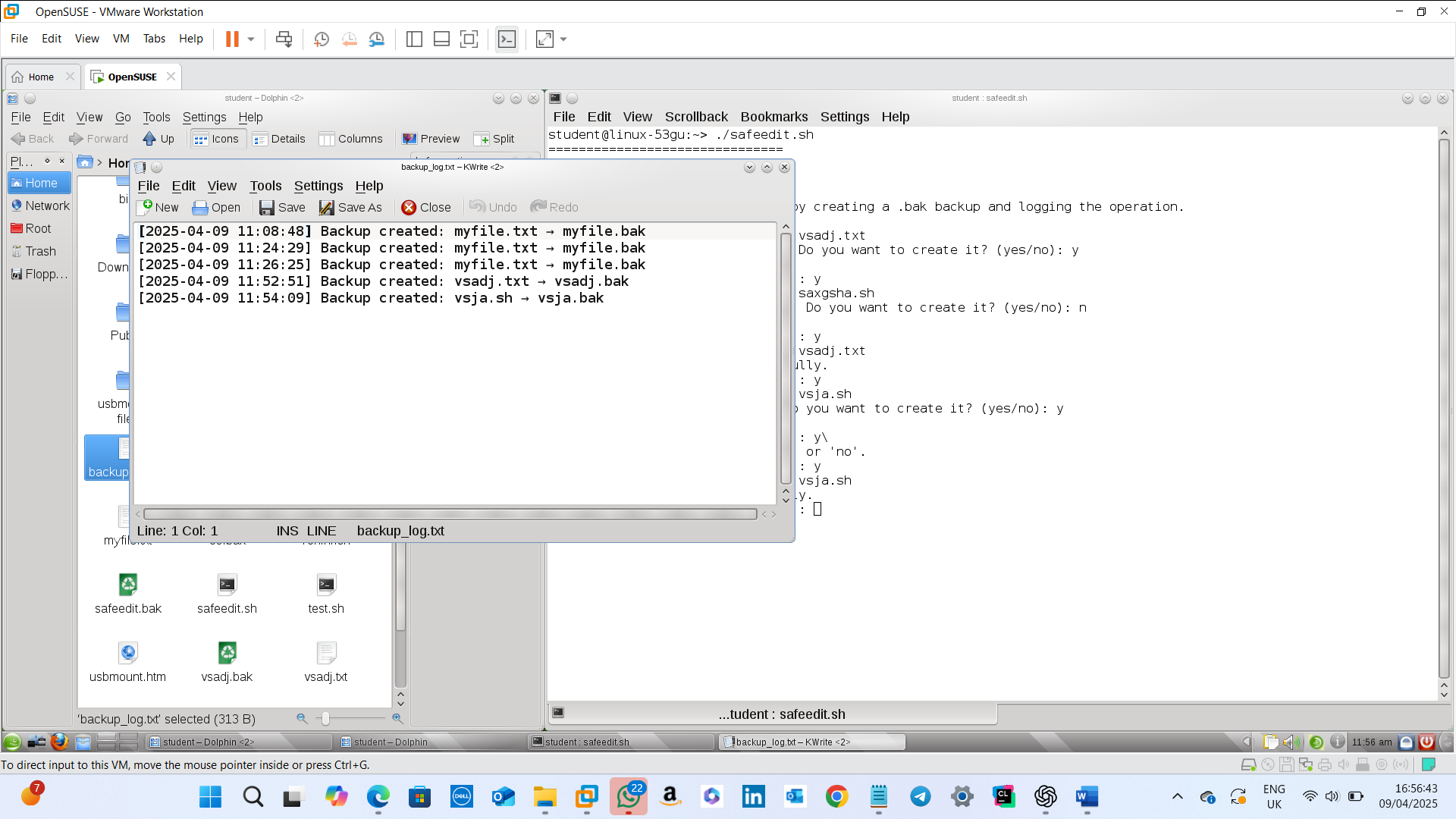
**Input:** Valid non-existing filename with .txt or .sh extension is entered.

**Result:** Confirmation is taken from the user to create a new file. If yes, a new file is created and if no, the creation is aborted.

**Test Case 5: Multiple Log Entries**

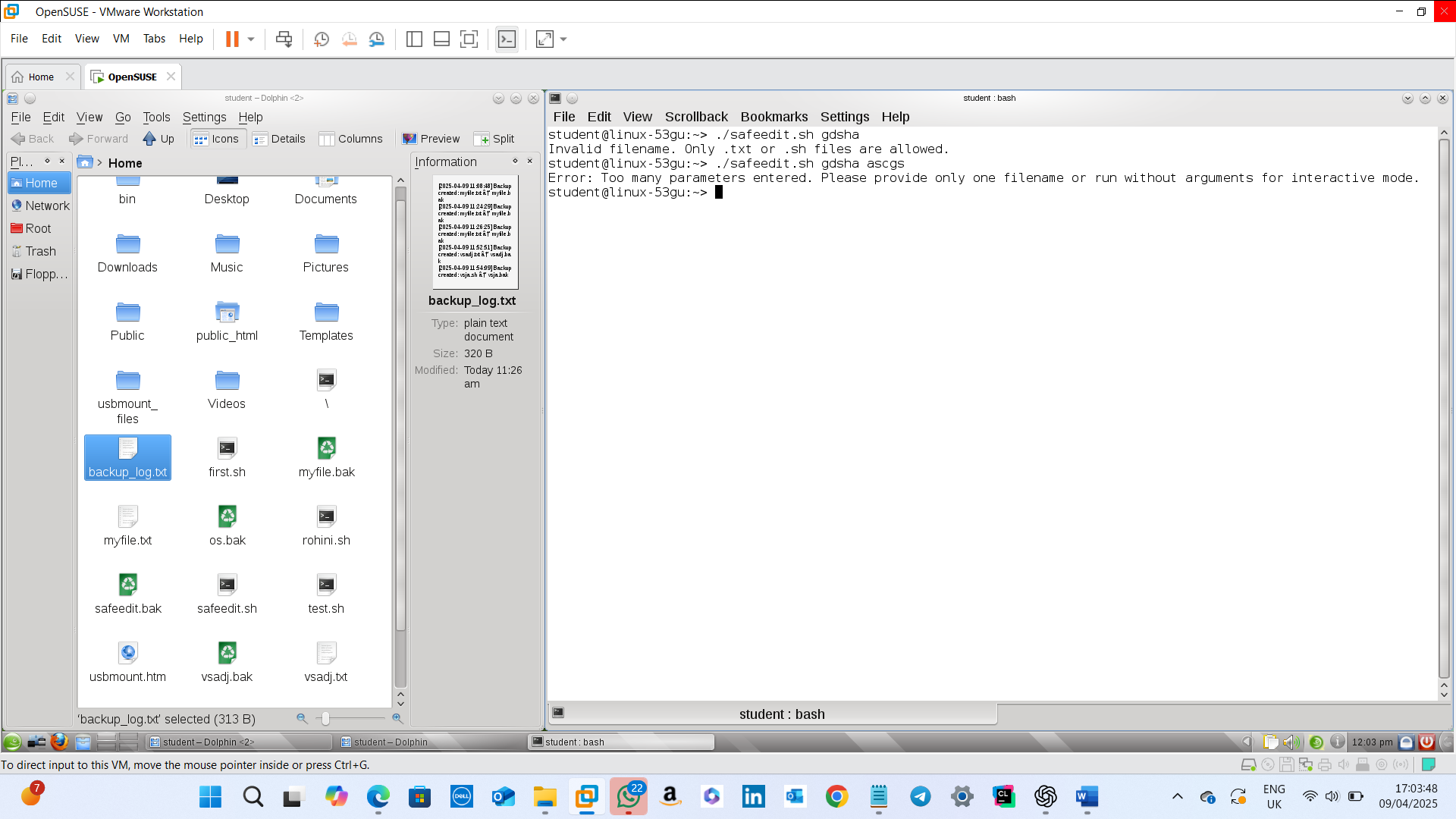


This will get removed after another entry to the same file!

**Input:** Perform several edits to generate multiple backups.  
**Result:** Log file retains only the last 5 entries, and older entries are removed.

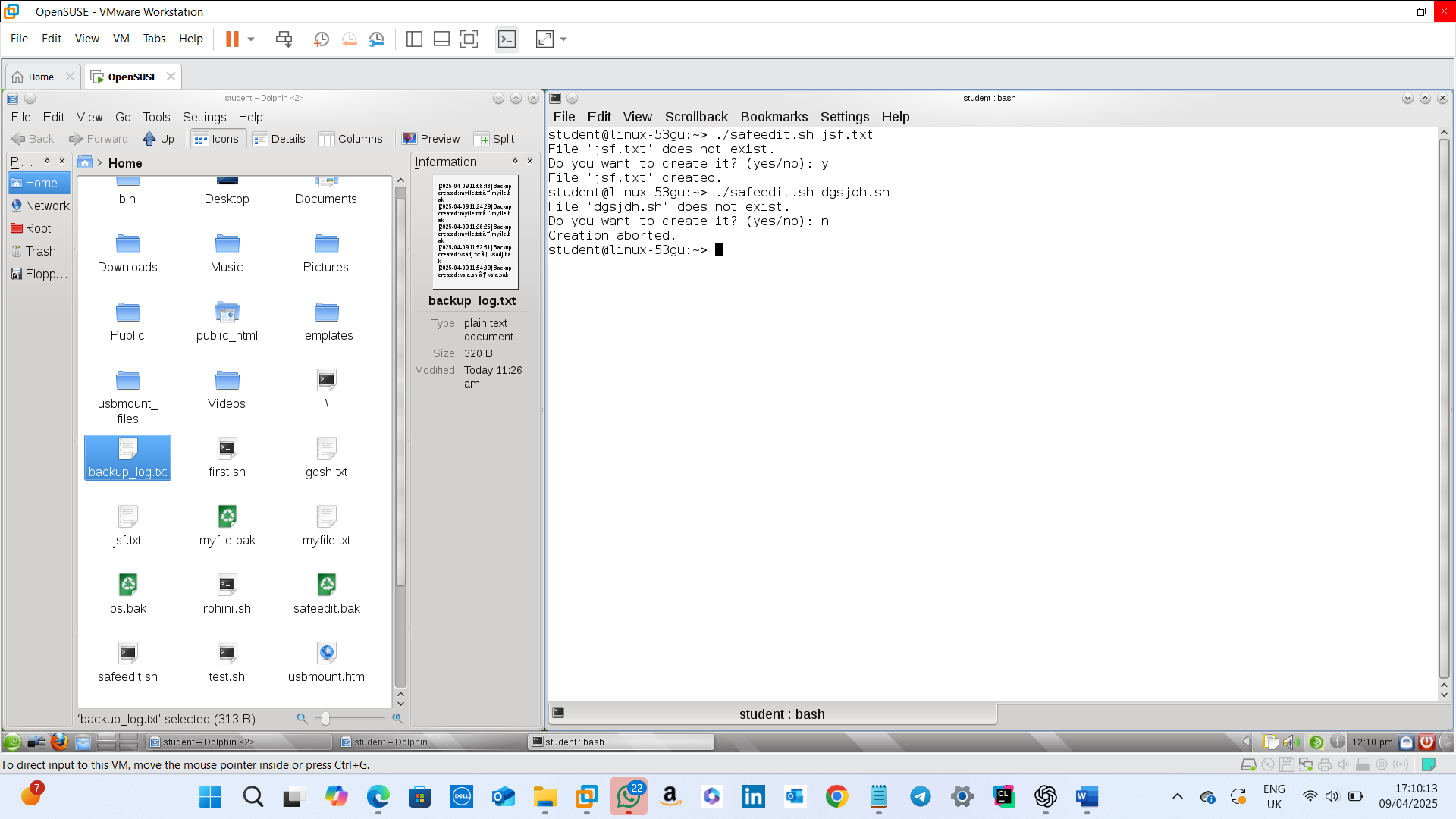
The oldest entry was removed and a latest entry was added into the backup\_log

**Test Case 6: Invalid filenames in Command-Line Mode**



**Input:** Filename is entered as argument.  
**Result:** If invalid filename is entered, that message is displayed and if too many arguments are passed, that message is displayed as error.

**Test Case 7: Non-existing Valid Filename in Command-Line Mode**



**Input:** Valid Filename which is not already existing is entered as argument.  
**Result:** Confirmation of creation is taken from the user ; if yes, new file is created and if no, the creation is cancelled.

**6. Conclusion**

In conclusion, this bash script allows the user to safely edit files. It offers a user-friendly environment on Unix-like systems. The error-handling built on this script makes this editing more user-friendly.

# Bibliography

# Stack Overflow. (n.d.) *How to trim whitespace in a batch script*. [online] Available at: <https://stackoverflow.com/questions/11434831/how-to-trim-whitespace-in-a-batch-script> [Accessed 5 Apr. 2025].

# Microsoft Docs. (n.d.) Batch file commands. [online] Available at: <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands> [Accessed 5 Apr. 2025].

1. SS64. (n.d.) CMD Commands. [online] Available at: <https://ss64.com/nt/> [Accessed 5 Apr. 2025].
2. GNU Operating System. (n.d.) Bash Reference Manual. [online] Available at: <https://www.gnu.org/software/bash/manual/> [Accessed 8 Apr. 2025].
3. Linuxize. (2019). How to Use Bash If Statement. [online] Available at: <https://linuxize.com/post/how-to-use-the-case-statement-in-bash/> [Accessed 8 Apr. 2025].
4. Stack Overflow. (n.d.). How to trim lines in bash using tail. [online forum] Available at: <https://stackoverflow.com/questions/19331837/bash-script-trimming-a-file-to-last-100-lines> [Accessed 8 Apr. 2025].
5. The Linux Command Line. (2012). A Complete Introduction. William Shotts. [Book] No Starch Press.s