

Amar Varpe ITTE57

Omkar ShelkeITTE58

Vaibhav Vavhal ITTE59

FileNest: Collaborative File Management System

# Introduction:

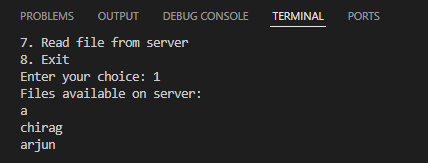
The File Management System is designed to facilitate efficient management of files stored on a server. It provides users with the ability to perform various file operations,

including creation, deletion, renaming, writing to, and reading from files. The system is implemented with both a command-line interface (CLI) and a graphical user interface (GUI), offering flexibility and ease of use to users.

# Features and their usage:

## Command-Line Interface (CLI):

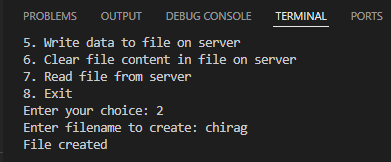
* + 1. **List Files:** Users can view a list of files available on the server.
       - **Usage:** Enter the option 1 :



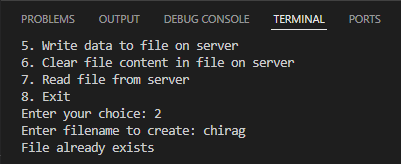


2

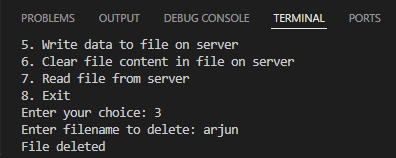
* + 1. **Create File:** Users can create a new file on the server.
       - **Usage:** Choose option 2, enter filename and a file by that name will be created.



If a file by that name already exists, it shows File already exists:



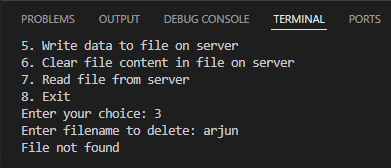
* + 1. **Delete File**: Users can delete an existing file from the server.
       - **Usage:** Choose the option 3, enter the filename, and if such a file exists, it will be deleted:



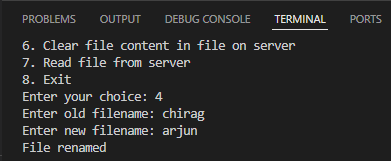


3

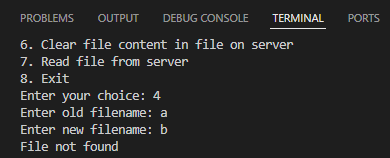
If such a file does not exist, then it shows file not found:



* + 1. **Rename File:** Users can rename an existing file on the server.
       - **Usage:** Choose the option 4, enter the old filename, new filename and the file will be renamed.



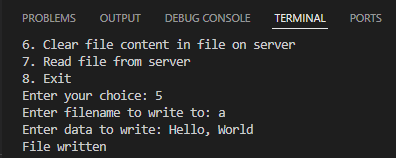
If no such file exists, it will show file not found.



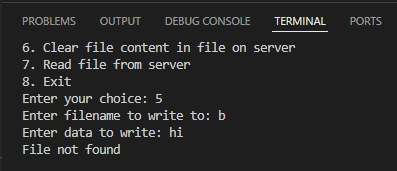


4

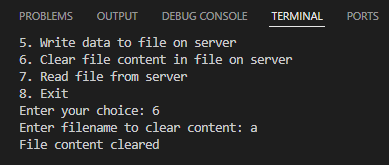
* + 1. **Write to File:** Users can write data to an existing file on the server.
       - **Usage:** Choose the option 5, enter the filename, enter data to write to the file and the data will be written/appended to the corresponding file.



If no such file exists, it shows file not found.



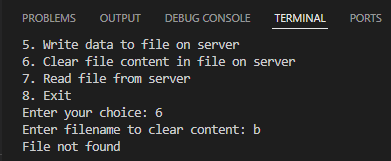
* + 1. **Clear File Content:** Users can clear the content of an existing file on the server.
       - **Usage:** Choose the option 6, enter the filename and it will clear all the file content.



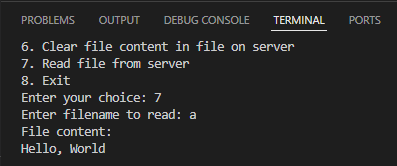


5

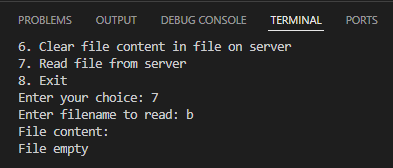
Similar to all the other operations, it shows file not found if the file doesn’t exist.



* + 1. **Read File**: Users can read the content of an existing file on the server.
       - **Usage:** Choose the option 7, enter the filename and it show the file content.



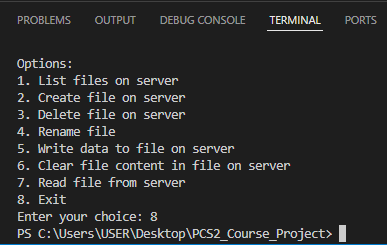
If the file is empty, it shows file empty.





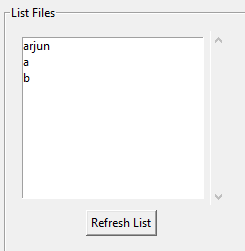
6

* + 1. **Exit:** Users can exit/end their connection with the server.
       - **Usage:** Choose the option 8, and the program will exit.



## Graphical User Interface (GUI):

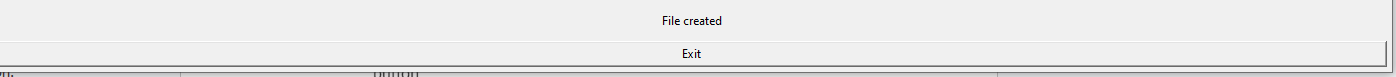
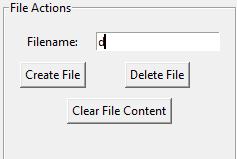
* + 1. **List Files:** Displays a list of files available on the server.
       - **Usage:** Click on the refresh list button to see the list of files in the scrollable text box above the button.





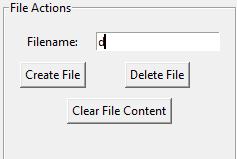
7

* + 1. **Create File:** Provides input field to specify a filename and button to create the file.
       - **Usage:** Enter the filename in the text box and click on the create file button.



If the file already exists, it will show file already exists above the exit button on the bottom of the window.

* + 1. **Delete File:** Allows users to select a file from the list and delete it.
       - **Usage:** Enter the filename in the text box and press the delete file button.



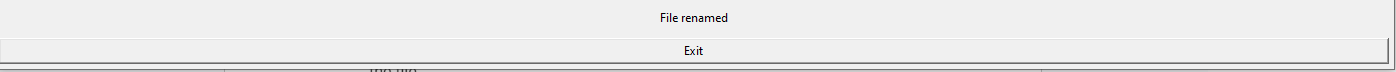
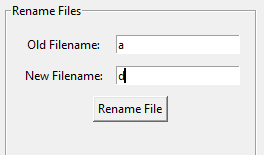
As we can see, it shows file deleted if it is successfully deleted.

If no such file exists, it will show, file not found.



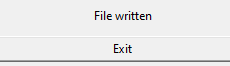
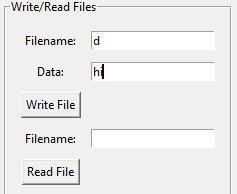
8

* + 1. **Rename File:** Users can enter the old and new filenames and click a button to rename the file.
       - **Usage:** Enter the old filename and the new filename in the text boxes provided and click on the rename file button,



As we can see, it shows file renamed. If no such file exists, it shows file not found.

* + 1. **Write to File:** Input fields for filename and data, with a button to write the data to the file.
       - **Usage:** Enter the filename and the data in the text boxes provided and click on the write file button.

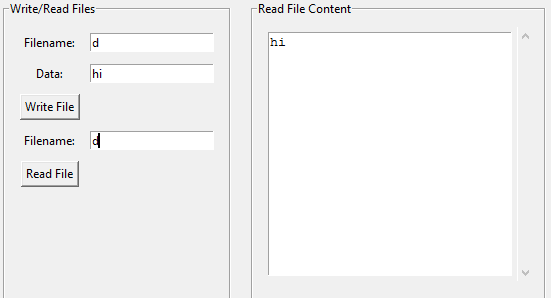




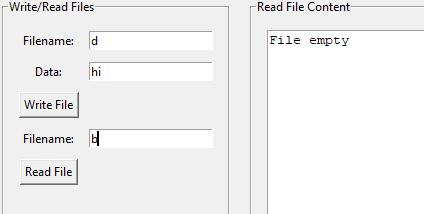
9

As we can see, it shows file written, if no such file exists it will show file not found.

* + 1. **Read File:** Input field for filename and a button to read and display the file content.
       - **Usage:** Enter the filename and click on the read file button.



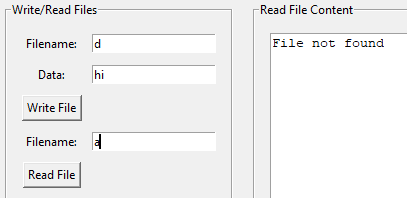
As we can see it shows the file content in the scrollable read file content text box. If the file is empty, it shows file empty.



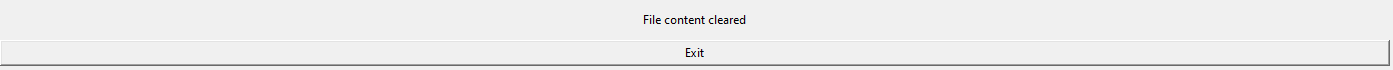
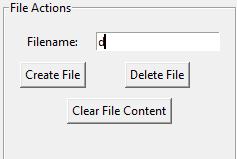


10

If no such file exists, it shows file not found in the read file content text box.



* + 1. **Clear File Content:** Allows users to clear the content of a file by specifying the filename.
       - **Usage:** Enter the filename in the text box and click on the clear file content button.

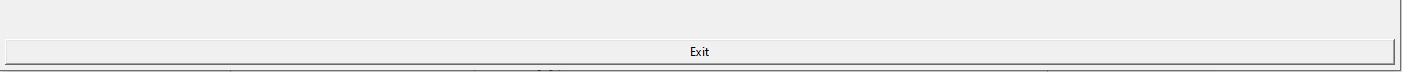


As we can see above, it shows file content cleared. If no such file exists, it shows file not found.



11

* + 1. **Exit:** Users can exit/end their connection with the server.
       - **Usage:** Click on the exit button on the bottom of the window and the window will close and the program exits.



# Implementation Details:

## Server Side (server.py):

* Implemented using Python's socket programming.
  + IP Address: The IP address where the server is hosted. 127.0.0.1 for

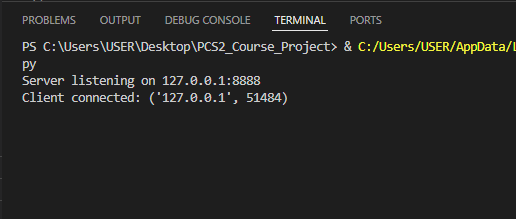
running the server and the client on the same machine, replace it with the router IP address to run server and client on different machines. Make

sure the server machine and the client machine are connected to the same network. The router IP address can be found in the Network/Wifi settings.

* + Port: The port number where the server is listening for incoming connections. It's set to 8888, but it can be any available port number. Ports are used to differentiate between different services running on the same machine.
  + Buffer Size: The size of the buffer used for sending and receiving data over the network. It's set to 4096 bytes in this project, meaning that data is sent and received in chunks of 4096 bytes.
  + Listen(5): 5 clients can connect to the server at a time and access the filesystem. The server creates a separate thread for each client.
* Listens for incoming connections and handles each client request in a separate thread.



12



* File system operations are encapsulated within the FileSystem class. This class encapsulates various file-related operations such as creation, deletion, renaming, writing, reading, and listing. Here's an explanation of how the file system is implemented:

1. Initialization:
   * The FileSystem class is initialized with an empty dictionary self.files to store file data. Each file is represented as a key-value pair, where the key is the filename and the value is the content of the file.
2. List Files:
   * The list\_files() method returns a list of filenames present in the file system. It extracts the keys from the self.files dictionary and returns them as a list.
3. Create File:
   * The create\_file(filename) method creates a new file with the specified filename. It checks if the filename already exists in the self.files

dictionary. If not, it adds an entry with an empty string as the

content and returns True to indicate success. If the filename already exists, it returns False.

1. Delete File:
   * The delete\_file(filename) method deletes a file with the specified

filename from the file system. It checks if the filename exists in the



13

self.files dictionary. If it does, it removes the entry corresponding

to the filename and returns True to indicate success. If the filename does not exist, it returns False.

1. Rename File:
   * The rename\_file(old\_filename, new\_filename) method renames a file

from old\_filename to new\_filename. It checks if the old\_filename exists in the self.files dictionary. If it does, it updates the key in the

dictionary with the new filename and returns "File renamed". If the

old\_filename does not exist, it returns "File not found".

1. Clear File:
   * The clear\_file(filename) method clears the content of a file with the specified filename. It checks if the filename exists in the self.files

dictionary. If it does, it sets the content of the file to an empty string and returns True to indicate success. If the filename does not exist, it returns False.

1. Write File:
   * The write\_file(filename, data) method writes data to a file with the specified filename. It checks if the filename exists in the self.files dictionary. If it does, it appends the data to the existing content of the file. If the filename does not exist, it returns False.
2. Read File:
   * The read\_file(filename) method reads the content of a file with the specified filename. It checks if the filename exists in the self.files

dictionary. If it does, it returns the content of the file. If the filename does not exist, it returns "File not found".

Overall, the FileSystem class provides a convenient interface for managing files within the FileNest application. It abstracts away the complexities of file manipulation and provides simple methods for interacting with files.

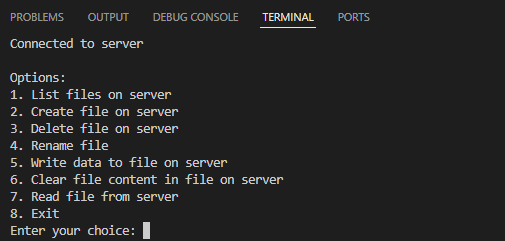


14

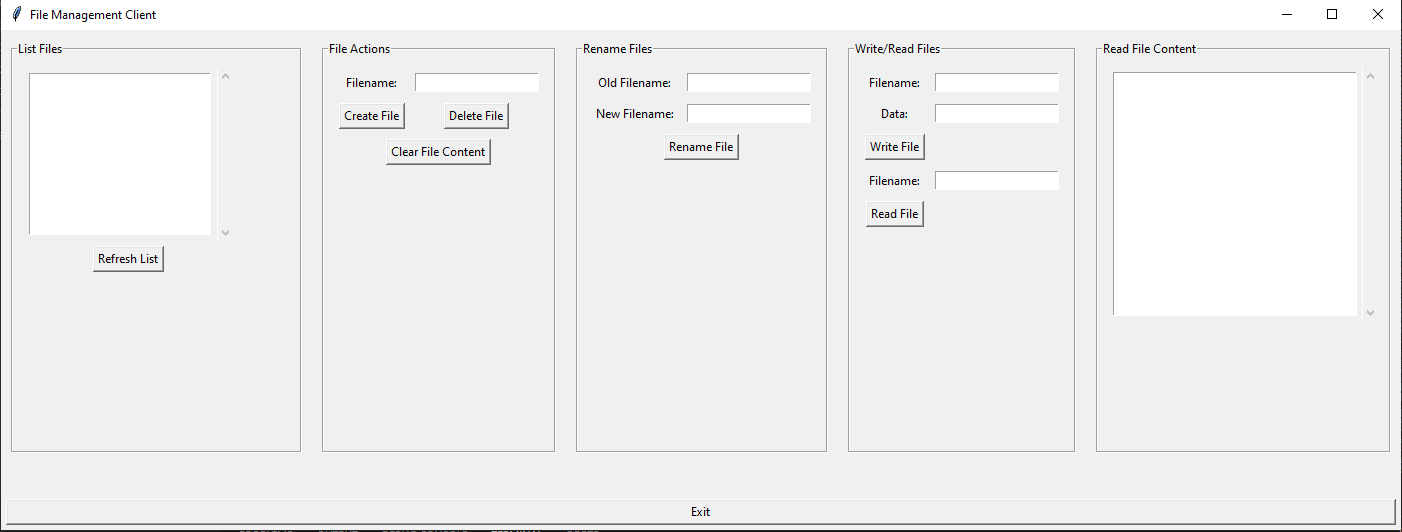
* Error handling for connection failures and invalid requests.

## Client Side (client.py & client\_GUI.py):

* CLI (client.py):
  + Sends requests to the server and displays responses in the command-line interface.
  + Menu-based interface for users to choose file operations.



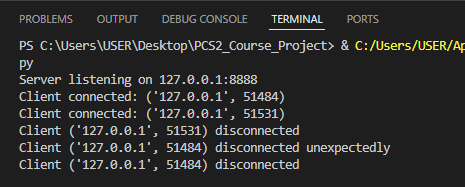
* GUI (client\_GUI.py):
  + Built using Tkinter library for Python.
  + Provides input fields and buttons for performing file operations.
  + Real-time updates for file lists and content display.



15

# Error Handling:

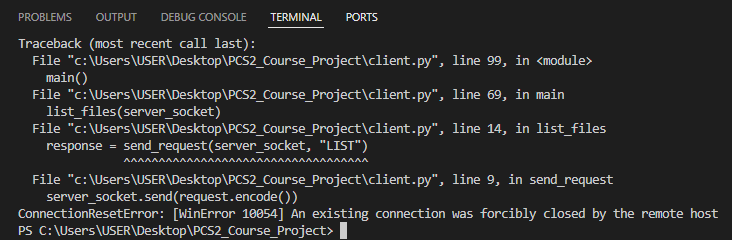
* Both server and client applications implement error handling mechanisms.
* Error messages are displayed to users to inform them of any issues encountered during file operations.
* Graceful handling of unexpected situations such as connection failures or invalid requests.
* When a client disconnects, the server shows this:



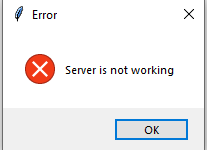


16

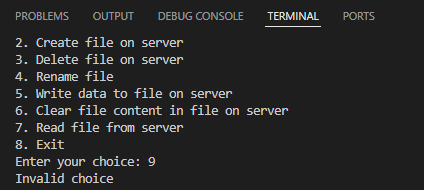
* When the server stops running and the CLI client tries an operation, it shows this error:



* When the server stops running and the client tries an operation, a message box is shown saying server not working for GUI client:



* If the CLI client inputs a choice other than (1-8) those which are shown:





17

# Future Enhancements that can be made:

* Enhanced Security: Implement user authentication and access control mechanisms.
* File Transfer: Allow users to upload and download files between client and server.
* Improved GUI: Enhance the GUI with additional features such as file browsing and drag-and-drop functionality.
* Real-Time Updates: Implement real-time updates in the GUI to reflect changes made to files on the server.
* File Versioning: Introduce version control for files to track changes and enable rollback to previous versions.

# Conclusion:

The Collaborative File Management System provides a comprehensive solution for managing files on a server. With its CLI and GUI interfaces, it caters to different user preferences and simplifies file operations. By implementing error handling, security measures, and future enhancements, the system can evolve into a robust and

user-friendly file management solution for various applications and environments.

1. **Group Members:**
2. B22AI051: Arjun Bhattad
3. B22AI056: Chirag Kumar