

FileNest: Collaborative File Management System

1. 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.

2. Features and their usage:

2.1 Command-Line Interface (CLI):

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

7. Read file from server
8. Exit
Enter your choice: 1
Files available on server:
a
chirag
arjun
```

2. 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.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

5. Write data to file on server
6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 2
Enter filename to create: chirag
File created
```

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

5. Write data to file on server
6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 2
Enter filename to create: chirag
File already exists
```

3. 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:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

5. Write data to file on server
6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 3
Enter filename to delete: arjun
File deleted
```

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

5. Write data to file on server
6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 3
Enter filename to delete: arjun
File not found
```

4. 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.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 4
Enter old filename: chirag
Enter new filename: arjun
File renamed
```

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 4
Enter old filename: a
Enter new filename: b
File not found
```

5. 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.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 5
Enter filename to write to: a
Enter data to write: Hello, World
File written
```

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 5
Enter filename to write to: b
Enter data to write: hi
File not found
```

6. 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.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

5. Write data to file on server
6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 6
Enter filename to clear content: a
File content cleared
```

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

5. Write data to file on server
6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 6
Enter filename to clear content: b
File not found
```

7. 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.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 7
Enter filename to read: a
File content:
Hello, World
```

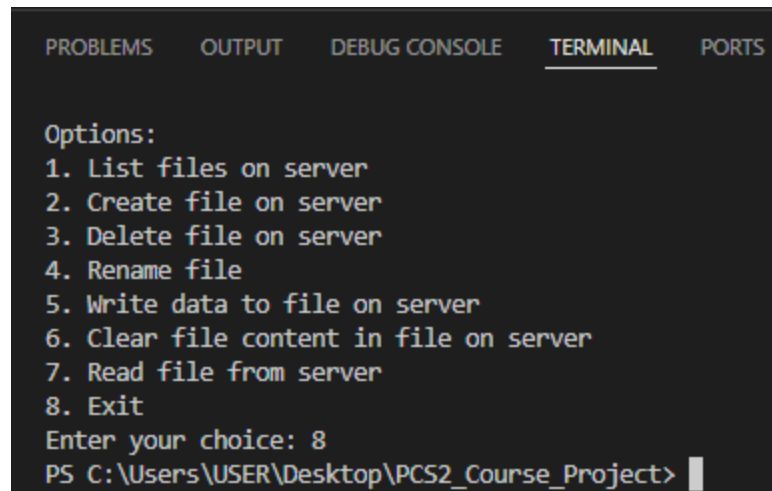
If the file is empty, it shows file empty.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 7
Enter filename to read: b
File content:
File empty
```

8. Exit: Users can exit/end their connection with the server.

- **Usage:** Choose the option 8, and the program will exit.



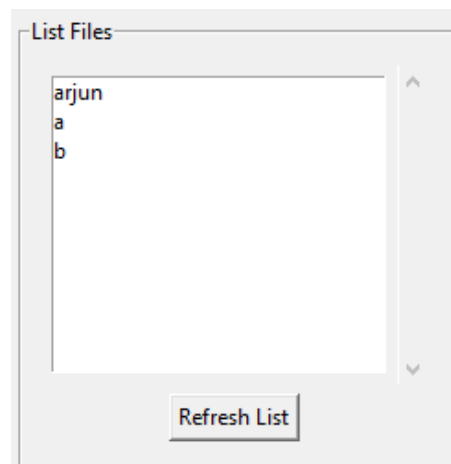
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Options:
1. List files on server
2. Create file on server
3. Delete file on server
4. Rename file
5. Write data to file on server
6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 8
PS C:\Users\USER\Desktop\PCS2_Course_Project>
```

2.2 Graphical User Interface (GUI):

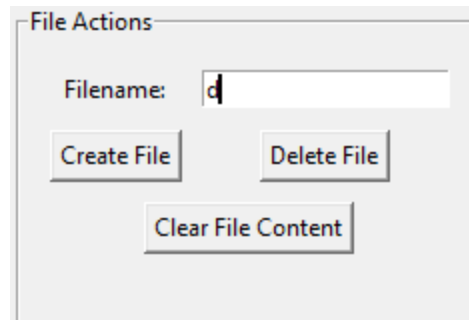
1. List Files: Displays a list of files available on the server.

- **Usage:** Click on the refresh list button to see the list of files in the scrollable text box above the button.

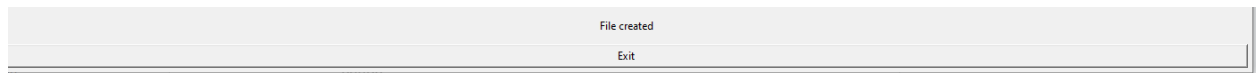


2. Create File: Provides input field to specify a filename and button to create the file.

- **Usage:** Enter the filename in the text box and click on the create file button.



The screenshot shows a dialog box titled "File Actions". It contains a "Filename:" label followed by a text input field containing the letter "d". Below the input field are three buttons: "Create File", "Delete File", and "Clear File Content". The "Create File" button is highlighted with a mouse cursor.

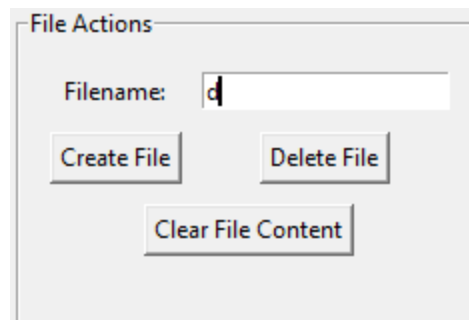


The screenshot shows a status bar with the text "File created" and an "Exit" button.

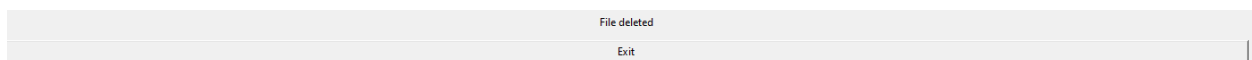
If the file already exists, it will show file already exists above the exit button on the bottom of the window.

3. 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.



The screenshot shows a dialog box titled "File Actions". It contains a "Filename:" label followed by a text input field containing the letter "d". Below the input field are three buttons: "Create File", "Delete File", and "Clear File Content". The "Delete File" button is highlighted with a mouse cursor.



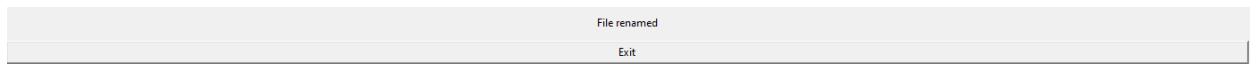
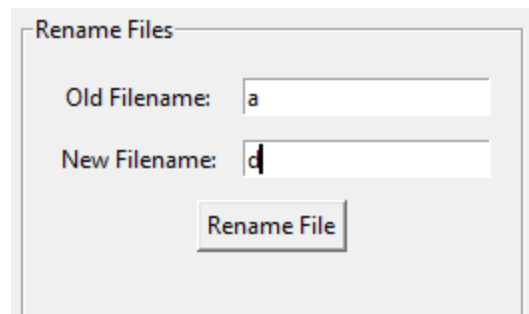
The screenshot shows a status bar with the text "File deleted" and an "Exit" button.

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

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

4. Rename File: Users can enter the old and new filenames and click a button to rename the file.

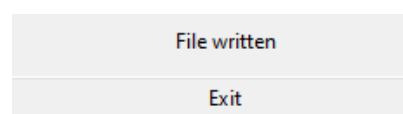
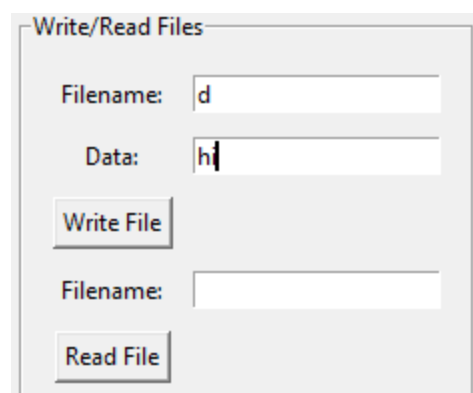
- **Usage:** Enter the old filename and the new filename in the text boxes provided and click on the rename file button,



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

5. Write to File: Input fields for filename and data, with a button to write the data to the file.

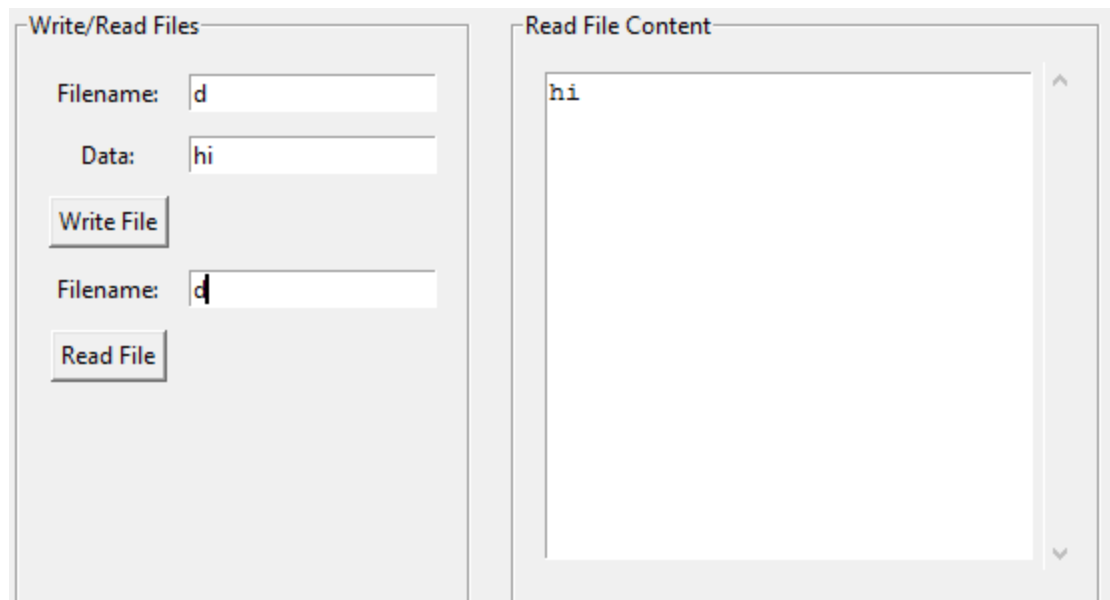
- **Usage:** Enter the filename and the data in the text boxes provided and click on the write file button.



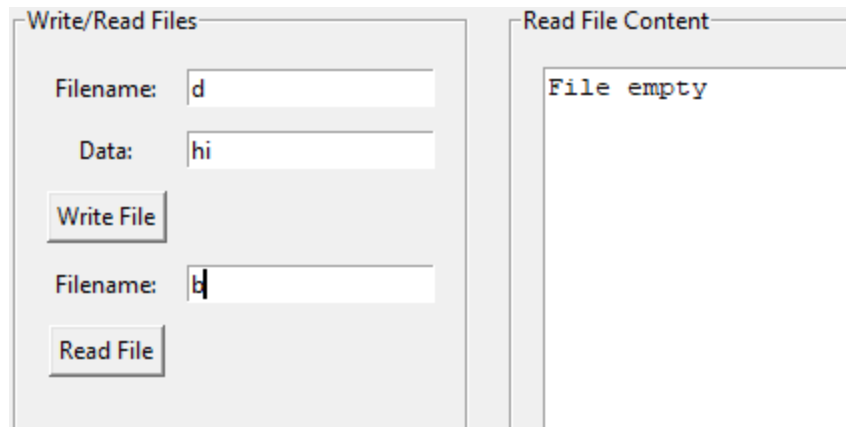
As we can see, it shows file written, if no such file exists it will show file not found.

6. Read File: Input field for filename and a button to read and display the file content.

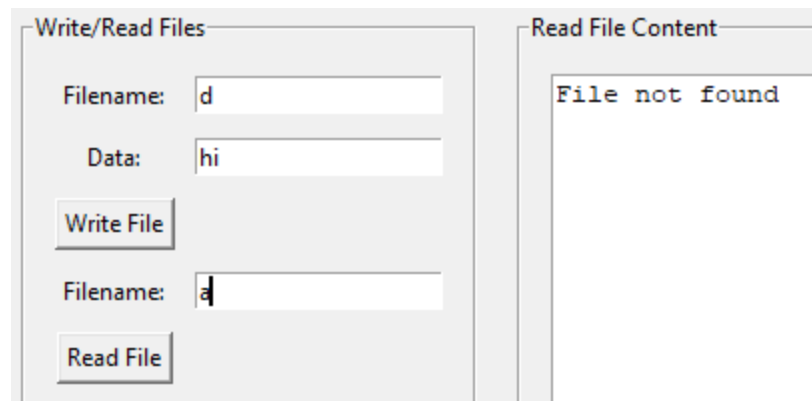
- **Usage:** Enter the filename and click on the read file button.



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.



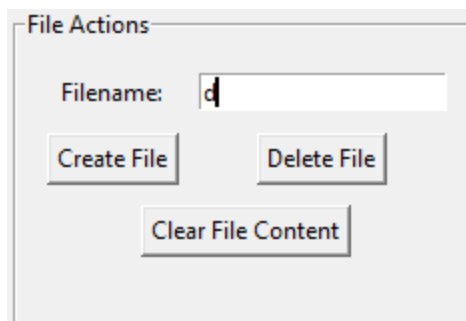
If no such file exists, it shows file not found in the read file content text box.



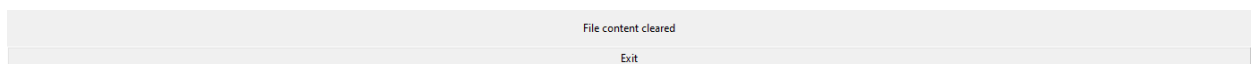
The screenshot shows a window with two panels. The left panel, titled 'Write/Read Files', contains two sets of controls. The top set has a 'Filename:' label and a text box with 'd', followed by a 'Data:' label and a text box with 'hi', and a 'Write File' button. The bottom set has a 'Filename:' label and a text box with 'a', and a 'Read File' button. The right panel, titled 'Read File Content', contains a text box displaying 'File not found'.

7. 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.



The screenshot shows a dialog box titled 'File Actions'. It contains a 'Filename:' label and a text box with 'd'. Below the text box are three buttons: 'Create File', 'Delete File', and 'Clear File Content'.



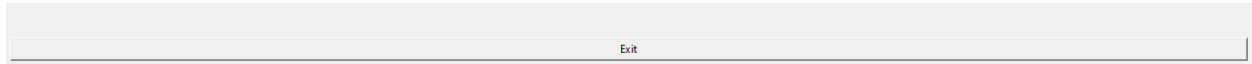
The screenshot shows a status bar with the text 'File content cleared' and an 'Exit' button.

As we can see above, it shows file content cleared.

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

8. Exit: Users can exit/end their connection with the server.

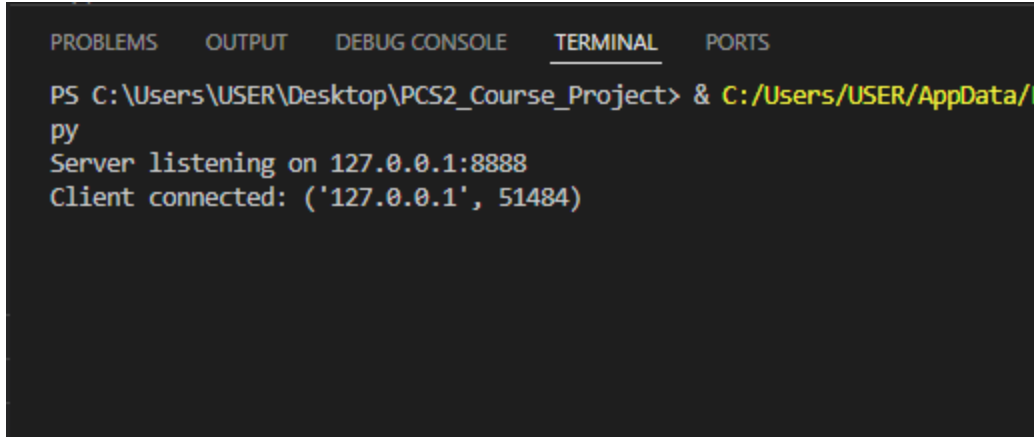
- **Usage:** Click on the exit button on the bottom of the window and the window will close and the program exits.



3. Implementation Details:

3.1 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.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\USER\Desktop\PCS2_Course_Project> & C:/Users/USER/AppData/L
py
Server listening on 127.0.0.1:8888
Client connected: ('127.0.0.1', 51484)
```

- 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`.
 4. 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

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.

5. 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".

6. 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.

7. 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.

8. Read File:

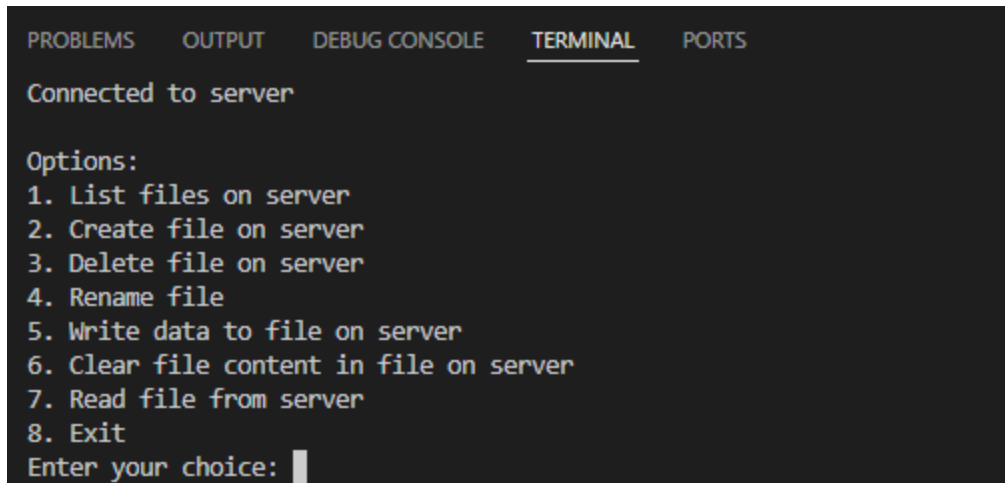
- The `read_file(filename)` method reads the content of a file with the specified filename. It checks if the filename exists in the `self.files` dictionary. If it does, it returns the content of the file. If the filename does not exist, it returns "File not found".

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.

- Error handling for connection failures and invalid requests.

3.2 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.

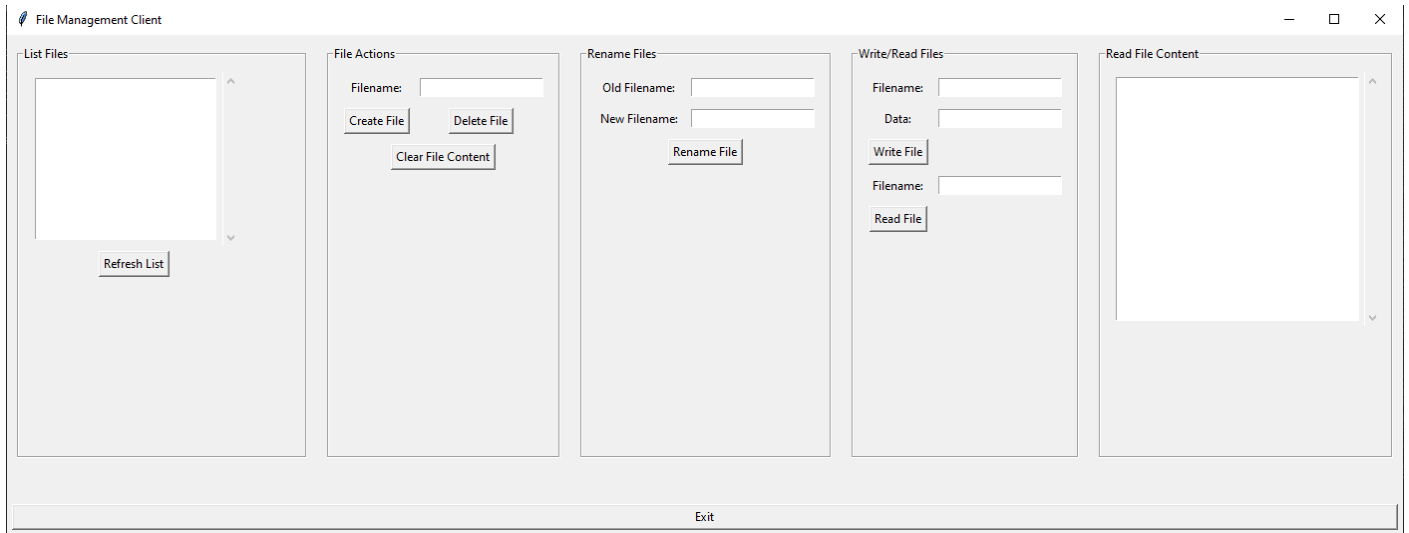


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Connected to server

Options:
1. List files on server
2. Create file on server
3. Delete file on server
4. Rename file
5. Write data to file on server
6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: █
```

- 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.



4. 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:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\USER\Desktop\PCS2_Course_Project> & C:/Users/USER/App
py
Server listening on 127.0.0.1:8888
Client connected: ('127.0.0.1', 51484)
Client connected: ('127.0.0.1', 51531)
Client ('127.0.0.1', 51531) disconnected
Client ('127.0.0.1', 51484) disconnected unexpectedly
Client ('127.0.0.1', 51484) disconnected

```

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

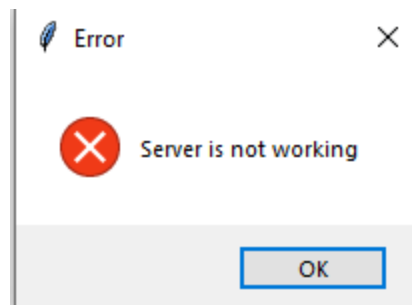
```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Traceback (most recent call last):
  File "c:\Users\USER\Desktop\PCS2_Course_Project\client.py", line 99, in <module>
    main()
  File "c:\Users\USER\Desktop\PCS2_Course_Project\client.py", line 69, in main
    list_files(server_socket)
  File "c:\Users\USER\Desktop\PCS2_Course_Project\client.py", line 14, in list_files
    response = send_request(server_socket, "LIST")
               ~~~~~
  File "c:\Users\USER\Desktop\PCS2_Course_Project\client.py", line 9, in send_request
    server_socket.send(request.encode())
ConnectionResetError: [WinError 10054] An existing connection was forcibly closed by the remote host
PS C:\Users\USER\Desktop\PCS2_Course_Project>

```

- 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:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

2. Create file on server
3. Delete file on server
4. Rename file
5. Write data to file on server
6. Clear file content in file on server
7. Read file from server
8. Exit
Enter your choice: 9
Invalid choice

```


5. 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.

6. 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.

7. Group Members:

1. B22AI051: Arjun Bhattad
2. B22AI056: Chirag Kumar