FILE EXPLORATION APPLICATION

High-Level and Low-Level Design

**Agenda**

1. Project Overview
2. High-Level Design
3. Low-Level Design
4. Conclusion

Project overview

**Objective:**

*Develop a console-based file explorer application in C++ that interfaces with the Linux operating system to manage files and directories.*

***Key Features:***

* *Listing directory contents*
* *Creating, renaming, and deleting files and directories*
* *Viewing file contents*
* *Navigating directories*

**HIGH LEVEL ARCHITECTURE:**

***1. System Overview***

*The File Explorer Application is a console-based tool developed in C++ that interacts with the Linux OS. It provides functionalities such as file listing, directory navigation, file manipulation (copy, move, delete, create), file search, and file permission management. The application will also have logging capabilities for both the client and server sides to record all actions.*

***2. Key Components***

* *FileManager: Manages all file operations like listing, creating, deleting, copying, and moving files.*
* *DirectoryManager: Handles directory-related operations such as navigating through directories and creating directories.*
* *SearchManager: Implements file search functionality within directories.*
* *PermissionManager: Manages file and directory permissions.*
* *Logger: Captures all console outputs and writes them to a log file, separate for the server and client.*
* *CommandParser: Parses user inputs and maps them to the respective operations.*
* *Main Application Loop: Orchestrates the overall flow by interacting with the different managers based on user input.*

***3. System Flow***

* *The application starts by displaying a command menu to the user.*
* *Based on user input, the corresponding command is parsed and executed by invoking functions from the appropriate manager (FileManager, DirectoryManager, etc.).*
* *The Logger logs each action to both the console and a log file.*
* *The application continues to run in a loop until the user exits***.**

LOW LEVEL ARCHITECTURE

*Low-Level Design delves into the detailed design of the system’s components. It specifies the logic, algorithms, data structures, and interfaces for each component or module identified in the High-Level Design. LLD provides a micro view of the system, focusing on the implementation details.*

***1. FileManager Class***

* ***Methods****:*
  + *listFiles(directoryPath): Lists files in the specified directory.*
  + *createFile(filePath): Creates a new file.*
  + *deleteFile(filePath): Deletes the specified file.*
  + *moveFile(sourcePath, destinationPath): Moves a file.*
  + *copyFile(sourcePath, destinationPath): Copies a file.*

***2. DirectoryManager Class***

* ***Methods****:*
  + *changeDirectory(newPath): Changes the current working directory.*
  + *createDirectory(directoryPath): Creates a new directory.*
  + *listDirectoryContents(directoryPath): Lists all contents in a directory.*

***3. SearchManager Class***

* ***Methods****:*
  + *searchFiles(directoryPath, fileName): Searches for files in the specified directory.*

***4. PermissionManager Class***

* ***Methods****:*
  + *setPermissions(filePath, permissions): Sets the permissions for a file or directory.*
  + *getPermissions(filePath): Retrieves the permissions for a file or directory.*

***5. Logger Class***

* ***Methods****:*
  + *logToFile(message): Writes the message to the log file.*
  + *logToConsole(message): Prints the message to the console.*
  + *log(message): Logs the message to both the console and the file.*

***6. CommandParser Class***

* ***Methods****:*
  + *parse(command): Parses the user input and calls the corresponding methods from FileManager, DirectoryManager, etc.*

*CONCLUSION*

* ***HLD****: Focuses on the system as a whole, providing an overview of the entire system architecture.*
* ***LLD****: Focuses on individual components or modules, providing detailed design specifications.*
* *Together, they ensure that the system is designed efficiently and effectively, facilitating successful development, deployment, and maintenance*.

FLOWCHART:

