***File Explorer Application***

**High Level Design:**

**Part 1(Components):**

1) Main Application-This is the entry point of our application.It intializes the environment and controls the flow between diffrent modules.

2) Directory Module-It handles operations related to directories such as listing , navigating and managing directory content.

3) File Module-It handles operations related to file such as creating , copying , moving , deleting and reading files.

4) Navigation Module-It manages the user's current location within the file system and handles navigation commands.

5) Search Module-This module implements functionality to search for files and directories based on user input.

6) Permission Module-It manages file and directory permissions.

7) Ui Module-Provides a console-based interface for user interaction.

**High Level Design Diagram:**

**Low Level Design:**

1) Components and Classes:

FileSystem:

Responsibilities: Handle directory and file operations.

Attributes:

currentPath: Current working directory.

Methods:

listFiles(): Lists files in the current directory.

changeDirectory(cd /path/to/directory): Changes the current directory.

createFile(touch filename): Creates a new file.

deleteFile(rm filename): Deletes a file.

copyFile(cp file1.txt newfile.txt,cp file1.txt /path/to/destination/): Copies a file.

moveFile(mv source\_file destination): Moves a file.

searchFile(find /path/to/search -name "filename"): Searches for a file in the current directory.

setPermissions(chmod [options] mode file): Sets file permissions.

777: Read, write, and execute for everyone.

755: Read, write, and execute for the owner; read and execute for group and others.

644: Read and write for the owner; read-only for group and others.

2) CommandParser:

Responsibilities: Parse user input and execute corresponding commands.

Attributes:

commands: List of available commands.

Methods:

parseCommand(const std::string& input): Parses user input and returns the corresponding action.

executeCommand(./filename): Executes the command.

3) FileExplorer:

Responsibilities: Interface between the user and FileSystem.

Attributes:

fileSystem: Instance of FileSystem.

commandParser: Instance of CommandParser.

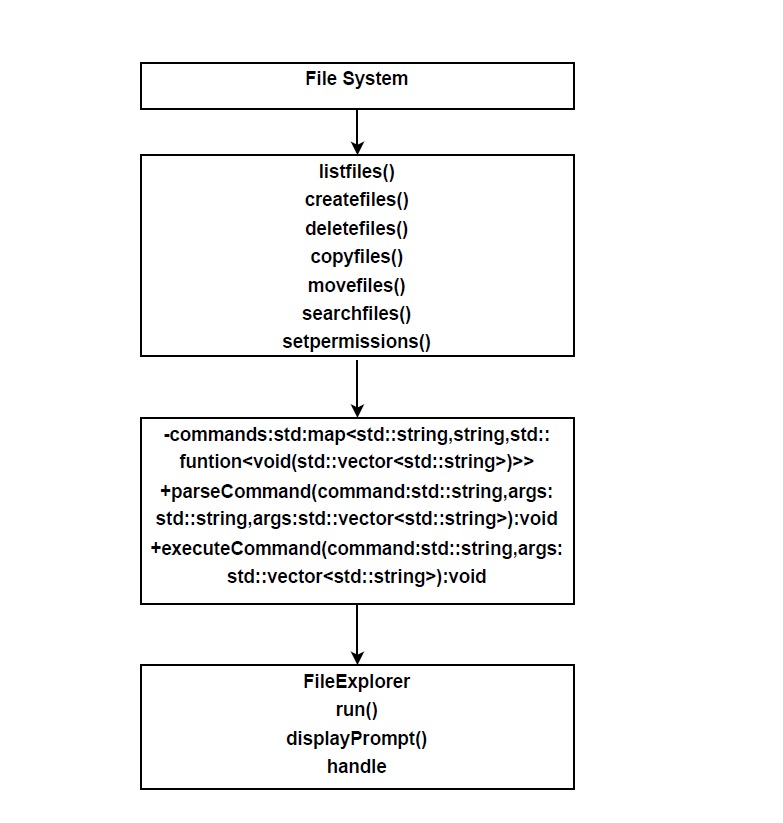
Methods:

run(): Main loop for interacting with the user.

displayPrompt(): Displays the command prompt.

handleUserInput(grep "handleUserInput" path/to/file.cpp): Handles user input and invokes corresponding methods on CommandParser.

**Class Diagram Of Low Level Design:**



**Activity Diagram Of Low Level Dsign:**



**Overall Project FlowChart:**