FILE EXPLORATION APPLICATION

High-Level and Low-Level Design

**Agenda**

1. Project Overview
2. High-Level Design
3. Low-Level Design
4. Dataflow
5. 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:**

***High-Level Design is the phase in the software development process where the overall architecture and structure of the system are defined. It outlines the system’s structure, identifying the main components, their relationships, and how they interact with each other. HLD provides a macro view of the system and serves as a bridge between the requirement specifications and the detailed design phase*.**

**Components:**

1. Main Application Module

* Initializes the application
* Manages the main loop

1. Command Handler Module

* Parses user commands
* Delegates tasks to appropriate modules

1. File System Module

* Performs file and directory operations

1. User Interface Module

* Handles user input and output

1. Error Handling Module

* Manages and reports errors

**Data Flow**

1. **User Input:** Command entered in the console
2. **Command Processing:** Command interpreted
3. **Operation Execution:** File System operations performed
4. **Error Handling:** Errors managed
5. **Result Display:** Results shown to the user

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.

**Main Application Module**

**Class: FileExplorer**

* **Attributes**:
  + CommandHandler
  + UserInterface ui
  + bool isRunning
* **Methods**:
  + void run(): Starts the main loop.
  + void stop(): Stops the main loop.

**Command Handler Module**

**Class: CommandHandler**

* **Attributes**:
  + FileSystem fileSystem
  + ErrorHandler errorHandler

**Methods**:

* + void parseAndExecute(const std::string& command): Parses the command and calls appropriate methods in the FileSystem and UserInterface modules.

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.