**1. Introduction**

* **Objective:** Discuss the development and implementation of the File Explorer Application, including requirements gathering, design considerations, and project timelines.
* **Overview:** The project aims to develop a console-based file explorer application in C++ for Linux, enabling users to manage files and directories through a command-line interface. Logging functionality will also be integrated.

**2. Project Requirements**

* **Functional Requirements:**
  + Listing files and directories.
  + Navigating through directories.
  + File operations (create, copy, move, delete).
  + Search functionality.
  + File permission management.
  + Logging of all actions with separate log files for server and client.
* **Non-Functional Requirements:**
  + Performance considerations, especially with large directories.
  + Security requirements to ensure appropriate file access.
  + Usability and portability across different Linux distributions.

**3. High-Level and Low-Level Design**

* **High-Level Design:**
  + Overview of application flow.
  + Main components: Command-line interface, file operations, logging system.
* **Low-Level Design:**
  + Detailed breakdown of each functionality.
  + Flowchart for directory navigation and file manipulation.
  + Logging mechanism for capturing both console output and file logs.

**4. Implementation Plan**

* **Development Environment:**
  + C++ with GCC/G++ compiler.
  + Linux OS (Ubuntu recommended).
  + Directory structure for source files, headers, binaries, and logs.
* **Task Breakdown by Day:**
  + **Day 1:** Set up development environment, basic file listing.
  + **Day 2:** Implement directory navigation.
  + **Day 3:** Add file manipulation capabilities.
  + **Day 4:** Implement file search functionality.
  + **Day 5:** Integrate file permission management.
  + **Continuous:** Implement and test logging functionality.

**5. Testing and QA**

* **Unit Testing:** Test individual modules for each operation.
* **Integration Testing:** Test combined operations.
* **System Testing:** Test the application in a full-scale environment.
* **Log Testing:** Ensure logs are properly created and maintained.

**6. Risks and Mitigation**

* **Potential Risks:**
  + Handling large directories efficiently.
  + Ensuring security during file operations.
* **Mitigation Strategies:**
  + Optimize code for performance.
  + Implement robust error handling and user access check.