Distributed Content Discovery & Inspection Utility
(DCDIU)

Design Documents - Version 0.0

Document Control:

Project Revision History

Date	Version	Author	Brief Description of Changes	Approver Signature
10-02-2026	Version 1.0	SAGNIK DEY TILAK KUMAR MERSHIKAJAYABAL RAKSHANA BABU	Initial implementation of core system structure with basic class separation and client-server communication setup.	
11-02-2026	Version 2.0	SAGNIK DEY TILAK KUMAR MERSHIKAJAYABAL RAKSHANA BABU	Enhanced system with user authentication mechanisms, improved process signaling, and better client session management.	
12-02-2026	Version 3.0	SAGNIK DEY TILAK KUMAR MERSHIKAJAYABAL RAKSHANA BABU	Implemented timestamp-based logging and comprehensive error handling features.	

Team Members

56124	SAGNIK DEY
56145	TILAK KUMAR S
58644	MERSHIKA JAYABAL
56141	RAKSHANA BABU S

Table of Contents

1) HIGH-LEVEL DESIGN

High Level Design

1. INTRODUCTION

1.1 Purpose

The purpose of this High-Level Design (HLD) document is to provide the necessary design details of the Distributed Content Discovery & Inspection Utility (DCDIU) so that the system can be implemented efficiently.
This document helps in identifying design inconsistencies prior to coding and serves as a reference for understanding high-level module interactions.

1.2 Scope

This document provides a comprehensive high-level design overview of DCDIU. It highlights the overall workflow and use cases involved in directory traversal, content discovery, and file inspection, and serves as an input to the Low-Level Design (LLD) document.

1.3 Definitions

1.3.1 DCDIU

Distributed Content Discovery & Inspection Utility (DCDIU) is a POSIX-compliant command-line utility designed to traverse directories, discover file contents based on patterns, and inspect file data.

1.3.2 User

A system administrator or user who interacts with DCDIU via a command-line interface**.**

1.3.3 POSIX File System

A standard-compliant file system that supports POSIX APIs for file and directory operations.

1.4 Overview

This HLD document is organized as follows:

- Section 1: Introduction – purpose, scope, definitions
- Section 2: General Description – constraints, tools, assumptions
- Section 3: Design Details – architecture, diagrams, interfaces

2. GENERAL DESCRIPTION

2.1 Product Perspective

DCDIU is a standalone command-line utility developed using C++ and POSIX APIs.The system is designed to help users audit and inspect file contents efficiently across local or distributed file systems.The system architecture is modular and layered, enabling scalability and maintainability.

2.2 Tools Used

- Linux / POSIX File System
- C++ Programming Language
- GNU Compiler (g++)

2.3 General Constraints

- The application must be CLI-based
- Only POSIX-compliant APIs must be used
- The system must operate in read-only mode

2.4 Assumptions

The assumptions relevant to the system are:

- User has required read permissions
- Target file system is accessible
- Linux terminal environment is available

2.5 Special Design Aspects

The system design is modular, allowing:

- Independent module testing
- Easy addition of new content filters
- Scalability for distributed extensions

3. DESIGN DETAILS

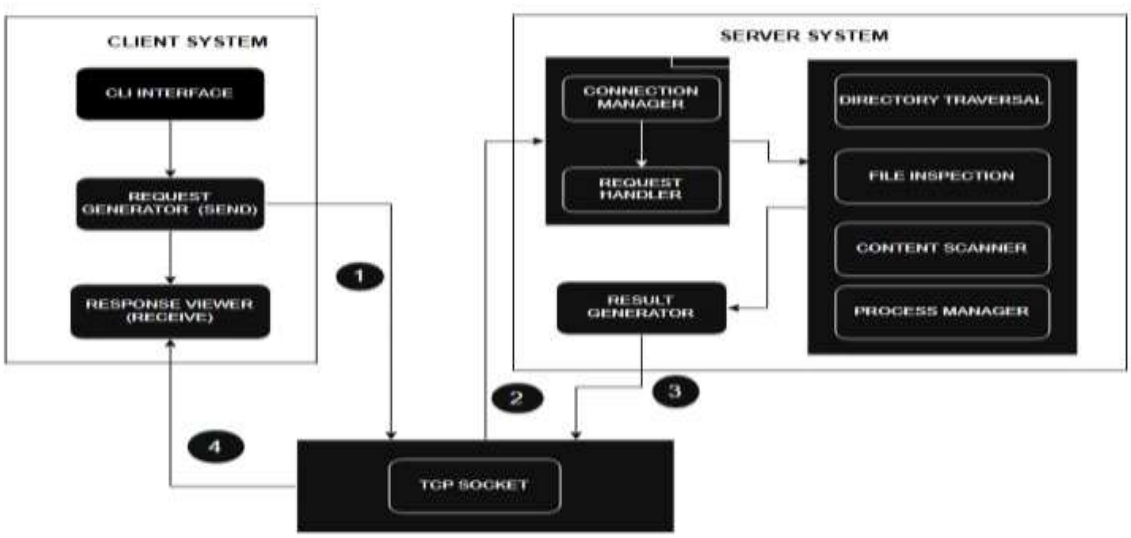
3.1 Main Design Features

The system is designed using a structured and modular approach to ensure reliability, scalability, and maintainability. The main design features include:

- Modular client–server architecture
- Process-based multi-client handling using fork()

- Secure authentication and access control mechanism
- Command-line based user interface
- TCP/IP-based communication protocol
- Directory traversal and content scanning modules
- File inspection and pattern matching functionality
- Session-based logging and audit trail generation
- Comprehensive error and exception handling
- Separation of concerns using independent modules

3.2 APPLICATION ARCHITECTURE

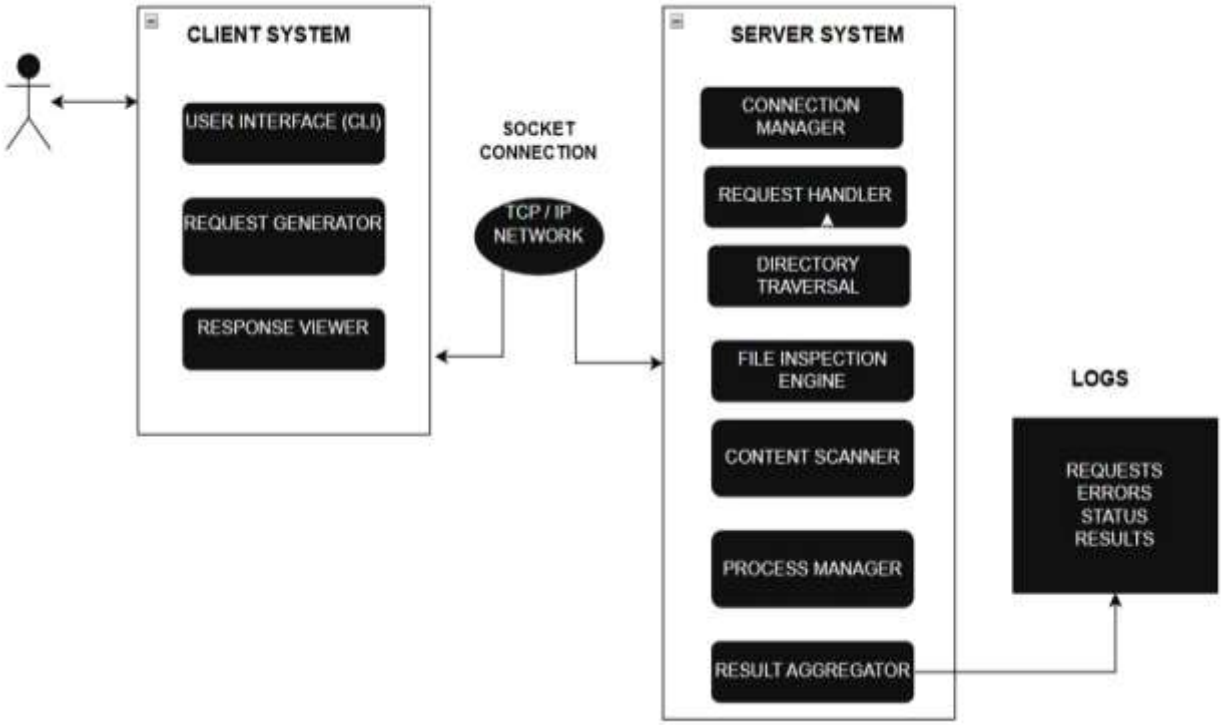


System Architecture Diagram Explanation (Point-wise)

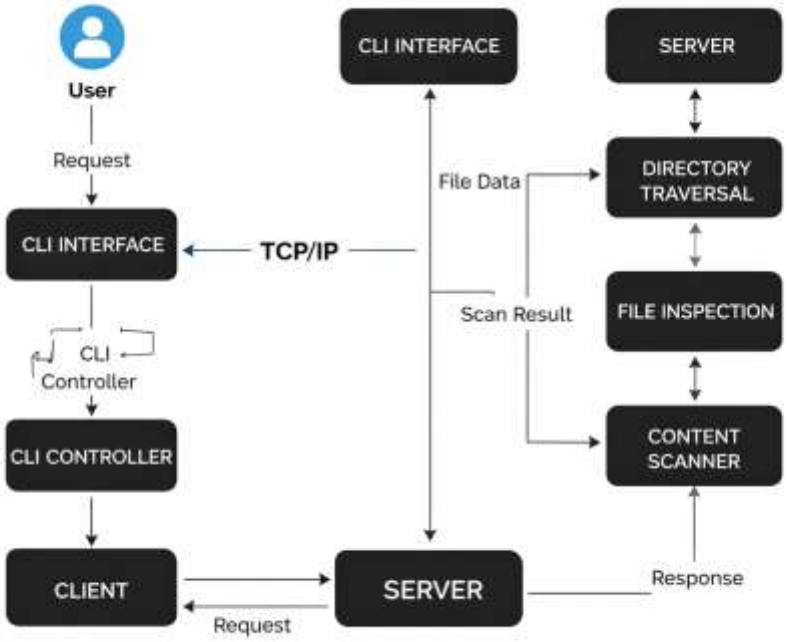
1. The system follows a client–server architecture using TCP/IP for reliable communication.
2. The client provides a Command-Line Interface (CLI) for users to enter commands and view results.
3. User commands are converted into requests by the Request Generator and sent to the server.
4. The TCP socket establishes and maintains communication between the client and server.
5. The server's Connection Manager accepts client connections and manages communication sessions.
6. The Request Handler processes and validates client requests before forwarding them to internal modules.
7. Processing modules such as Directory Traversal, File Inspection, and Content Scanner analyze the requested data.
8. The Result Generator formats the processed output and sends it back to the client for display.

.

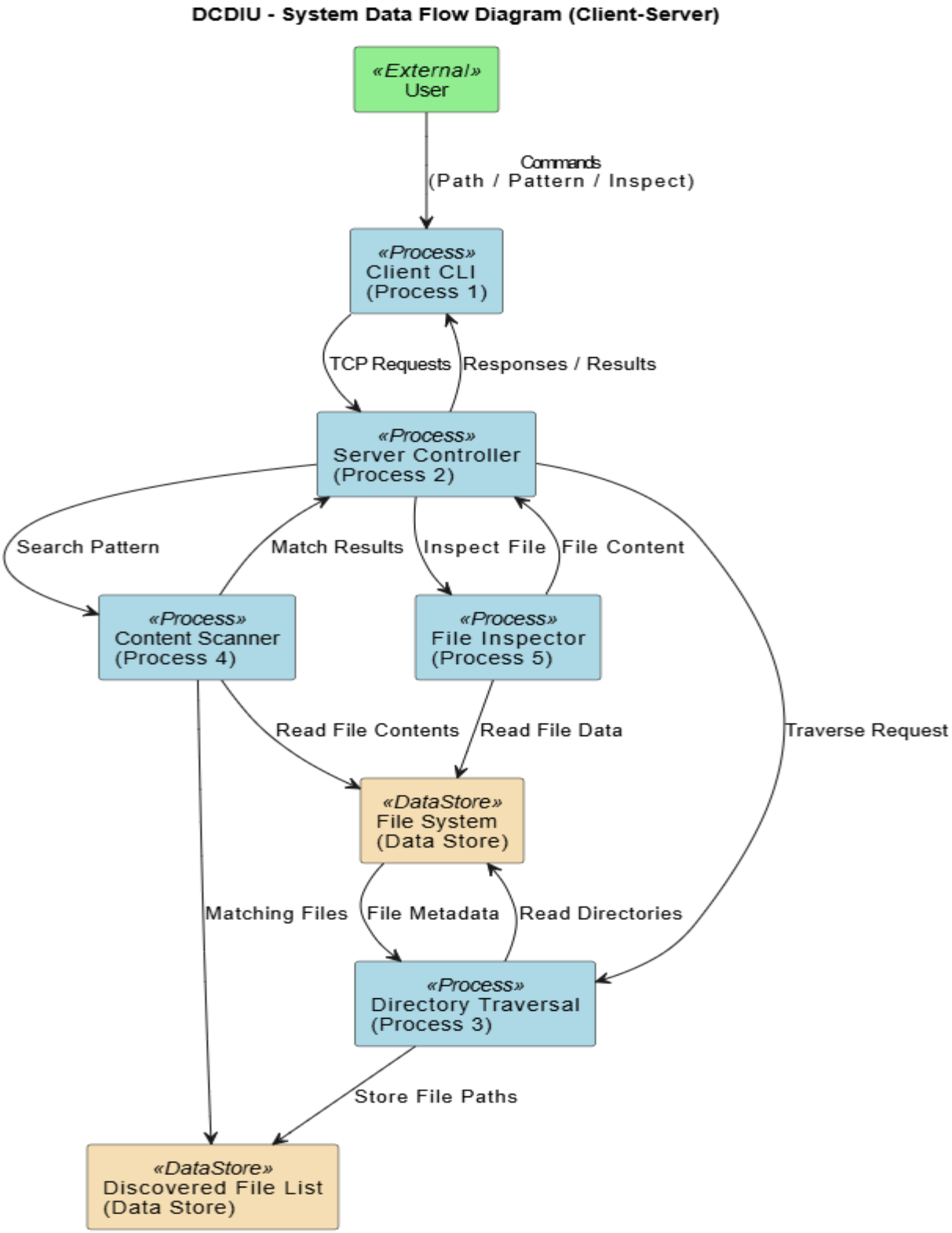
3.2 .1 System Architecture



3.2.2 System Interfaces



3.3 DATA FLOW DIAGRAM



3.4 Files

DCDIU reads files from the local file system during directory traversal, content scanning, and file inspection operations. The system generates auxiliary files such as files.txt for temporary storage and users.txt for credential management. Log files are created for each client session. The system does not modify, delete, or rename user files.

3.5 User Interface

The system runs on Linux-based desktop environments.

- A Command Line Interface (CLI) is provided for user interaction.
- Clients enter commands such as TRAVERSE, SEARCH, INSPECT, and EXIT.
- Authentication prompts are displayed for secure login.

3.6 Error Handling

Error handling is implemented using conditional checks and system call validations. The system detects and handles errors such as:

- File not found
- Permission denied
- Invalid command input
- Socket communication failures
- File read/write failures
- Authentication failures

All major errors are logged for debugging and auditing purposes.

3.7 Help

Help is provided through:

- Project documentation
- Command-line prompts
- Standard usage instructions
- Error messages and feedback

3.8 Performance

The system provides efficient directory traversal and content scanning using optimized recursive algorithms. It supports multiple concurrent clients using process-based concurrency and maintains acceptable response time for large directory structures.

3.9 Security

- Secure user authentication using salted SHA-256 hashing
- Read-only access to inspected files
- No modification of user files
- Operating system-based permission enforcement
- Session-based access control

3.10 Reliability

DCDIU ensures stable execution through proper resource management and error handling. Client disconnections and unexpected failures are handled gracefully and recorded in log files**.**

3.11 Maintainability

The modular design ensures:

- Easy debugging
- Clear separation of functionality
- Easy enhancements
- Minimal maintenance effort

3.12 Portability

- The system is portable across POSIX-compliant operating systems such as Linux and UNIX.
- It uses standard C++ and POSIX socket APIs.

3.13 Reusability

The system modules are designed for reuse in similar client–server applications. Efficient resource utilization ensures minimal CPU and memory overhead without persistent background processes.

3.14 Application compatibility

DCDIU is compatible with standard Linux file systems and POSIX environments. It supports common directory structures and file formats.

3.15 Resource utilization

The system uses moderate CPU and memory resources. Process-based concurrency ensures isolated execution without excessive resource consumption.

3.16 Major Classes and characteristics

- Server – Manages client connections and process creation
- ClientHandler – Handles authentication and client sessions
- DirectoryTraverser – Performs recursive directory scanning
- ContentScanner – Searches files for specific patterns
- FileInspector – Displays file contents
- Logger – Manages session logs
- Authentication Module – Handles secure login and credential storage

Low Level Design Document

4. Introduction

The aim of this document is to gather, analyze, and provide an in-depth insight into the complete Distributed Content Discovery & Inspection Utility (DCDIU) by defining the problem statement and solution approach in detail.The intended audience includes all stakeholders involved in the development and evaluation of the system. These include system administrators, developers, testers, project managers, and evaluators.This document provides the detailed low-level design of DCDIU, explaining internal module behavior, process flow, data handling, and system interactions at a granular level.

4.1Purpose

This Software Requirements Specification (SRS) document defines the functional and non-functional requirements for the Distributed Content Discovery & Inspection Utility (DCDIU) implemented using a client–server architecture.The purpose of this document is to provide a clear and structured description of the system's behavior, interfaces, constraints, and operating environment. It is intended for:

- Developers implementing the client and server components
- System architects designing the client–server interaction
- Instructors and evaluators assessing the project
- Testers responsible for validation and verification

This document serves as a reference throughout the design, implementation, testing, and maintenance phases of the system lifecycle.

4.2Document Conventions

The document uses the "Times New Roman" font with bold for the heading. The main heading size is 18. The Sub-heading size is 14. The content under the is of 12 points size. All the spacing are normal/default spacing of MS Word.

4.3 Intended Audience and Reading Suggestions

This document is intended for:

- Software Architects
- Software Developers
- System Engineers

- Quality Assurance Engineers
- Project Managers and Evaluators

Readers are expected to have a basic understanding of:

- Client-Server Architecture
- Networking Fundamentals
- Operating Systems
- File Handling in C++
- Multi-threaded Programming

4.4 References

The following authoritative references have been used to support the architectural design, client-server communication, file system operations, error handling, and security concepts defined in the Directory and Content Detection and Inspection Utility (DCDIU) system.

1.POSIX Standard – The Open Group Base Specifications
https://pubs.opengroup.org/onlinepubs/9699919799/
(Official POSIX documentation covering directory handling, file I/O, and system calls like opendir(), readdir(), stat(), open(), read().)

2. Linux Manual Pages (man7.org)
https://man7.org/linux/man-pages/
(Authoritative Linux reference for system calls like open(), read(), stat(), and socket APIs.)

3. C++ Reference – File and String Handling
https://en.cppreference.com/w/cpp/filesystem
(Official C++ reference for file systems and string operations like std::string::find().)

4. Beej's Guide to Network Programming
https://beej.us/guide/bgnet/
(Clear explanation of TCP socket programming used in your client-server communication.)

5. Linux System Programming – Michael Kerrisk
https://man7.org/tlpi/
(Advanced reference for low-level Linux programming, including file descriptors and system calls.)

5. Detailed System Design

This section describes the internal design and working structure of the Distributed Content Discovery and Inspection Utility (DCDIU). The system is designed with emphasis on modularity, scalability, maintainability, and reliability. Each functional component is developed as an independent module, enabling easy debugging, testing, and future enhancements. The design also ensures secure communication, efficient processing, and stable multi-client support.

5.1 Design Descriptions

The detailed design of DCDIU is organized into the following major components:

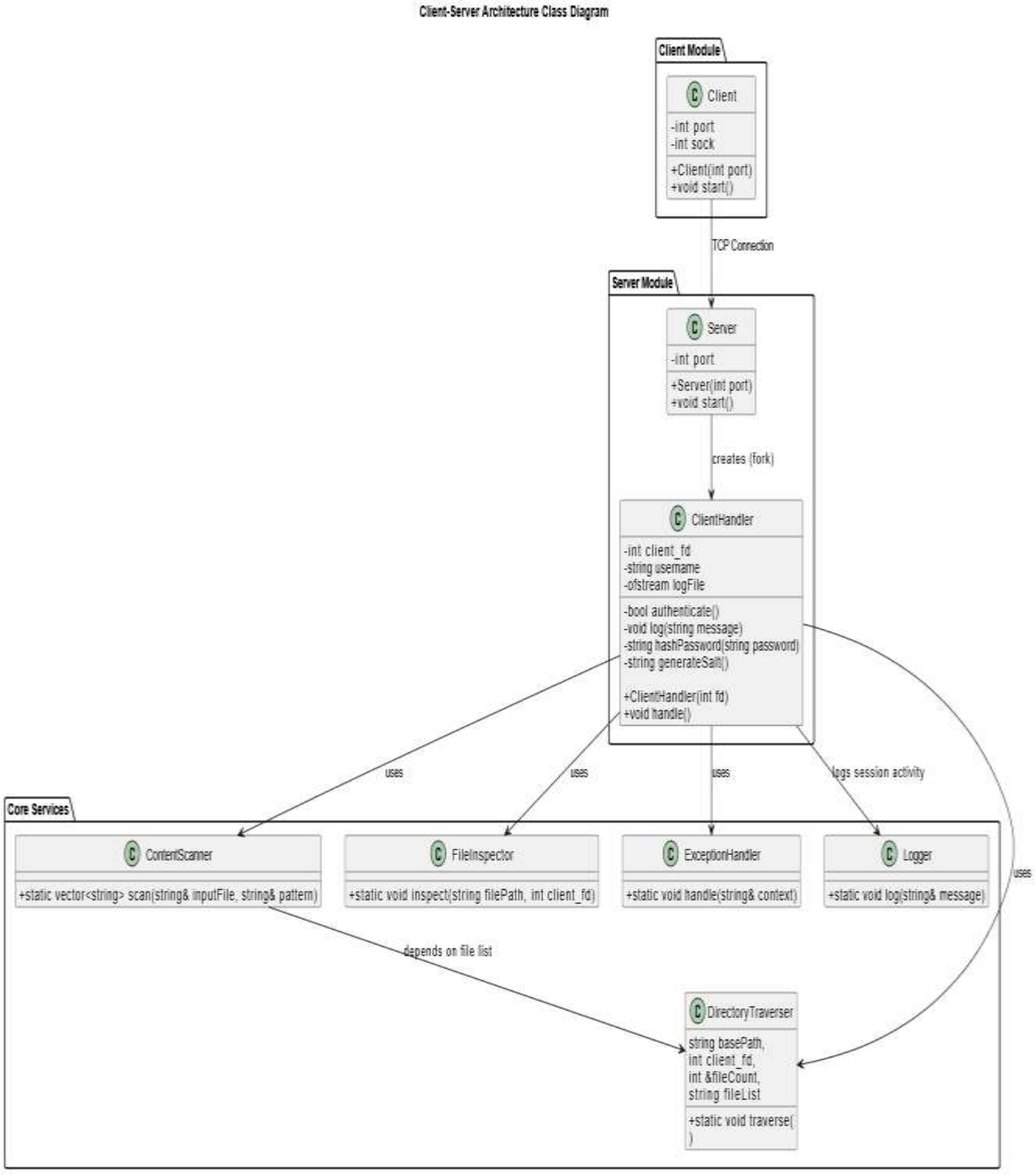
1. System Architecture
   Describes the client–server model based on TCP/IP communication. The server uses process-based concurrency (fork()) to handle multiple clients simultaneously, while each client interacts with the system through a command-line interface.
2. User Interface Design
   Defines the command-line interface used for user interaction. It includes authentication prompts, command input handling, and response display mechanisms for TRAVERSE, SEARCH, INSPECT, and EXIT operations.
3. File and Data Handling
   Explains the management of user files, temporary files (files.txt), credential storage (users.txt), and session log files. The system ensures read-only access to user data and secure storage of authentication information.
4. Process Flow and Automation
   Describes the sequence of operations from client connection and authentication to command execution and session termination. It includes automated handling of requests, response generation, logging, and resource cleanup.

System Mechanisms:

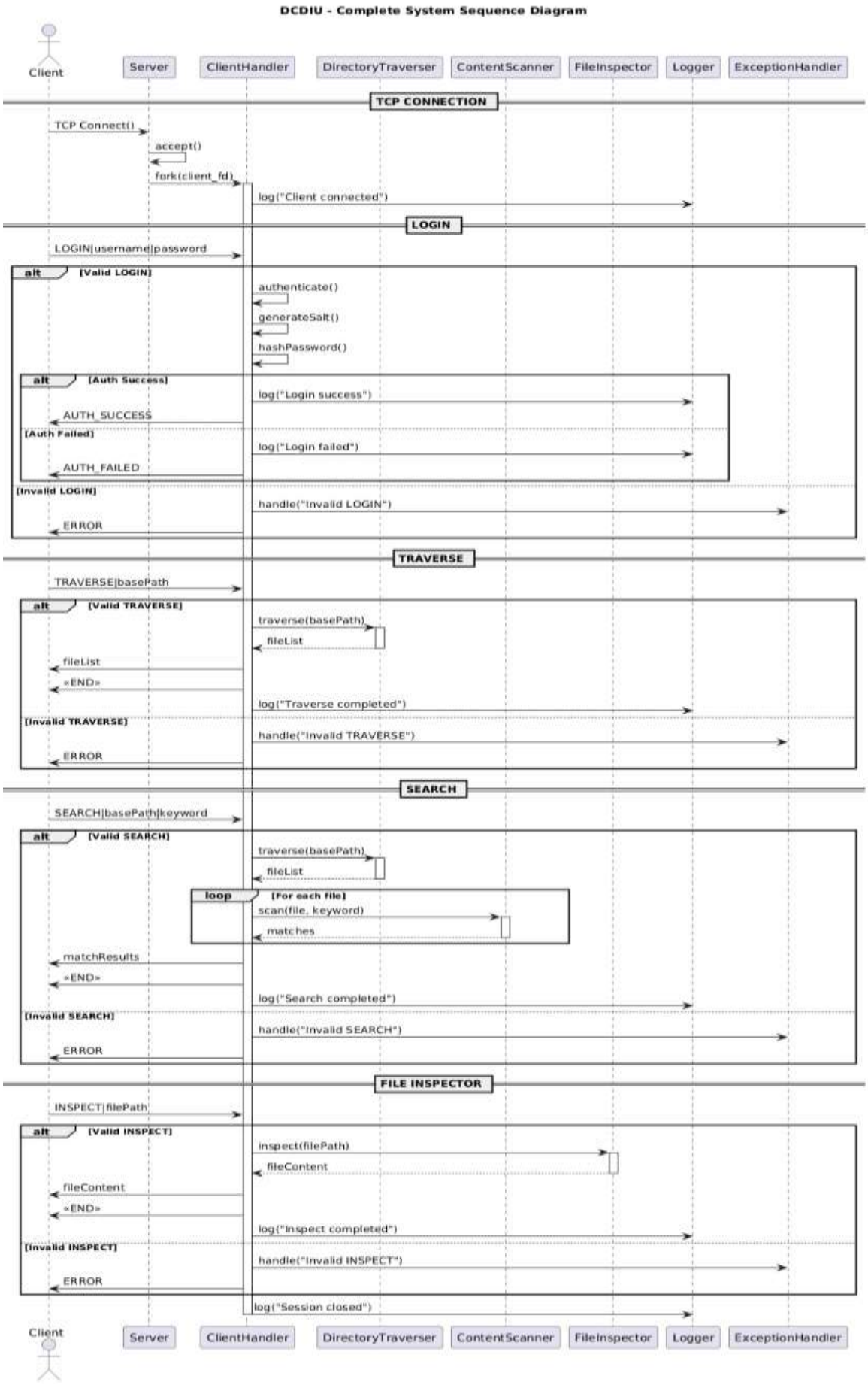
- The system requires users to log in using a valid username and password before accessing any services.
- Passwords are securely stored using salted SHA-256 hashing, ensuring that plain-text passwords are never saved.
- Only authenticated users are allowed to execute system commands and access server resources.
- The system operates in read-only mode and does not modify, delete, or rename user files.
- Operating system-level file permissions are enforced to prevent unauthorized file access.
- Each user session is logged to maintain security records and support monitoring and auditing.

5.2Class Diagram

The following Class Diagram represents the internal structure and relationships of the DCDIU Client-Server System. It illustrates the major classes, their attributes, methods, and interactions within the system. The diagram shows the separation between the client and server modules and explains how the server creates individual client handlers using process-based concurrency

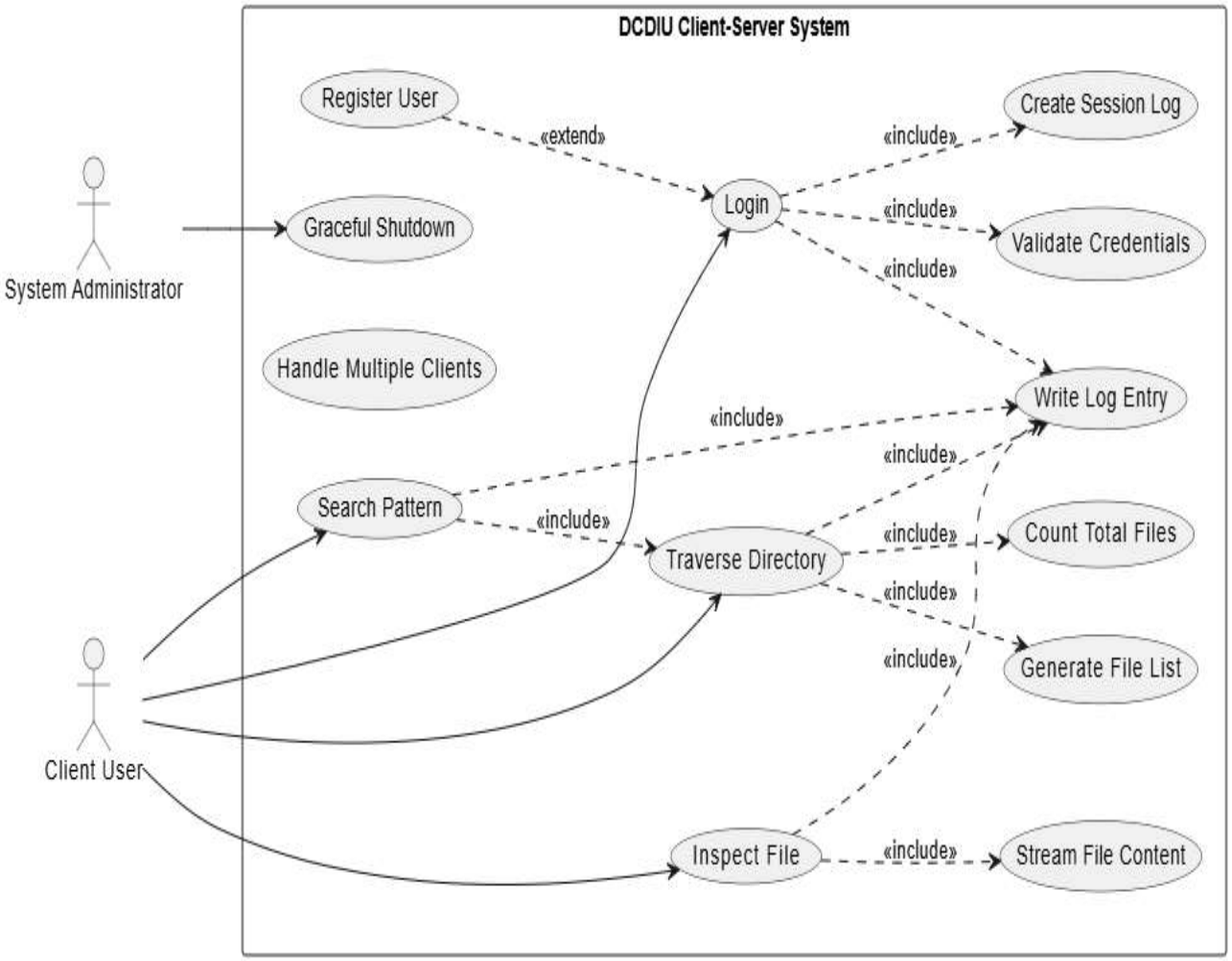


5.3 Sequence Diagram



5.4 Use Case Diagram

The following Use Case Diagram represents the functional behavior of the DCDIU Client-Server System. It illustrates the interactions between different users and the system, including the System Administrator and Client User. The diagram shows how users perform various operations such as user registration, login, directory traversal, pattern searching, file inspection, and session management.



5.5 Design and Implementation Constraints

The following constraints apply to the system:

- The system is implemented using the C++ programming language.
- Only POSIX-compliant system calls and socket APIs are used.
- The system operates primarily in read-only mode for user files.
- No graphical user interface is provided; interaction is limited to CLI.
- The system is designed to run on Linux and UNIX-based platforms.
- Process-based concurrency is used instead of threading.
- These constraints ensure portability, simplicity, and compatibility with standard operating environments.

5.6 User Interface

The user interface of DCDIU consists of:

- A desktop or Linux machine with terminal access
- A Command Line Interface (CLI)

All user interactions are performed using text-based commands. The interface provides authentication prompts, command input options, and result displays. Users interact with the system using commands such as TRAVERSE, SEARCH, INSPECT, and EXIT.

The CLI design ensures simplicity, low resource usage, and ease of deployment.

6.Security

DCDIU implements secure authentication and basic authorization mechanisms to protect system access. Security is enforced through both application-level controls and operating system permissions.

The system provides the following security features:

- User authentication using username and password
- Secure password storage using salted SHA-256 hashing
- Session-based access control after successful login
- Read-only access to user files
- No modification, deletion, or renaming of files
- Enforcement of OS-level file permissions
- Per-session logging for auditing purposes
- These security measures ensure that only authorized users can access system services and that the integrity of the host file system is preserved.