# High-Level Design & Low-Level Design

Cloud++

|  |  |
| --- | --- |
| Version No. | 1.00 |

|  |
| --- |
| **TEAM MEMBERS** |

|  |  |
| --- | --- |
| **EMPLOYEE ID** | **EMPLOYEE NAME** |
| 293956 | Abhijith Jose |
| 293959 | Ebi Shima Paul |
| 293945 | Nisha Budanuru Krishna |
| 293957 | Yasser Sillan Hydroos |

**HIGH LEVEL DESIGN**

**TABLE OF CONTENTS**

1. Introduction
   1. Abstract 5
   2. Intended Audience 5
   3. Acronyms/Abbreviations 5
   4. Project Purpose 5
   5. Key Project Objectives 6
   6. Project Scope and Limitation 6
      1. In scope
      2. Out of Scope
   7. System Overview 6
   8. Assumptions, Dependencies & Constraints 7
2. Design Overview
   1. Architecture Diagram 8
   2. Data Flow Diagram 10
      1. Level 0 Diagram
      2. Level 1 Diagram
   3. Sequence Diagram 11
   4. Use Case Diagram 12
   5. Performance 13
   6. Security 13
   7. Reliability 13
   8. Maintainability 13
3. Modular Description
   1. Client Layer 13
   2. Server Layer 14
   3. Storage Layer 14
   4. Admin and Utility Layer 14
4. Environment Description
   1. Hardware Requirements 15
   2. Software Requirements 15
5. Error Handling

**LOW LEVEL DESIGN**

1. Introduction
   1. Description 16
   2. Purpose 16
   3. References 16
2. Detailed System Design
   1. Design Description 16
   2. Class Diagram 17
   3. Activity Diagram 18
3. Demo 19

**Introduction**

**HIGH LEVEL DESIGN**

* 1. **Abstract**

The Cloud++ is a cloud storage project that is designed to offer a secure and efficient platform, that helps the users to store, manage, share files remotely. This project incorporates essential features within it, this includes user authentication, session management, file deletion and restoration. Cloud++ ensures a smooth and reliable data handling and ensures data integrity and user convenience.

* 1. **Intended Audience**

This document is intended for software developers, project managers and architects involved in the design, implementation, and review of the Cloud++. It provides both high-level overview and in-depth technical reference for future development, enhancement and maintenance.

* 1. **Acronyms/Abbreviations**
* CLI: Command Line Interface
* HLD: High-Level Design
* LLD: Low-Level Design
* API: Application Programming Interface
* OS: Operating System
* GUI: Graphical User Interface
  1. **Project Purpose**

The Cloud++ project aims to create a secure, scalable, reliable, and a user-friendly cloud storage platform that provide features to the users to manage, share and store files remotely, supporting robust authentication, quota management, file operations and efficient data organizations.

* 1. **Key Project Objectives**
* It provides the user with a strong user authentication and session management.
* Supports reliable file upload, download, deletion, restoration, and sharing.
* Enforce user quotas and secure file access.
* Modularize the system for maintainability and extensibility.
* Optimize for security, concurrency, and performance.
  1. **Project Scope and Limitation**
     + 1. **In Scope**
* User authentication, user registration, login, logout, and password management.
* File Operations: upload, download, list, delete, restore, and share.
* Storage Management with inbox and trash for each user(clients)/category.
* CLI clients for user interaction.
* Quota enforcement and management, and logging of operations.
  + - 1. **Out Of Scope**
* Real-time collaborative editing.
* Integration with third party cloud APIs.
* Native mobile or web GUI
  1. **System Overview**

The Cloud++ is a comprehensive system designed to provide the users/clients with a secure platform for managing the data in a cloud environment, hence enabling the users to perform essential operations such as user authentication, file uploads and downloads, quota management, and secure file sharing. This also incorporates deletion, restoration, and trash management.

The architecture of Cloud++ is modular, separating the system into distinctive layers and components respectively to ensure clarity, maintainability and scalability:

* + - Client Layer:

This acts as the user interface, enabling the users to interact with the system through command line.

* + - Application Layer:

This encapsulates the core logic. It is organized into modules responsible for user authentication and session management, file upload/download handling, quota management and file sharing mechanisms.

* + - Storage Layer:

This maintains the organized storage of user data across various directories, such as inbox and trash, hence ensuring efficient data retrieval.

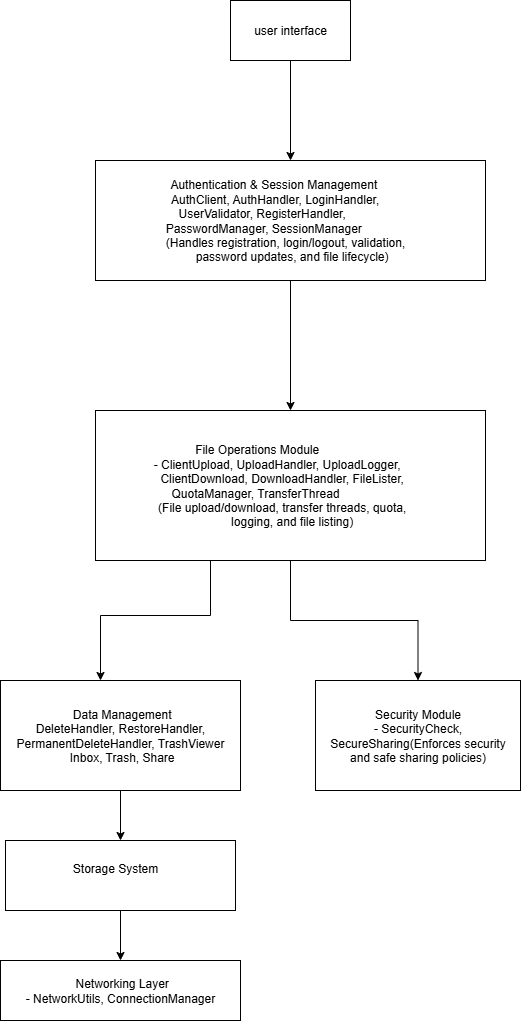
* + - Storage Management:

This supports other background functionality like file-listing, transfer threading, and logging, helping to enhance the reliability and performance of the data operations.

The design of Cloud++ emphasizes the separation of modularity thereby allowing independent development, testing and future enhancements.

* 1. **Assumptions, Dependencies & Constraints**
* Developed in C++.
* Uses TCP/IP for client-server communication.
* File system-based storage.
* Command-line clients only.

1. **Design Overview**
   1. **Architecture Diagram**

****

* Client Layer:

It serves as the entry point for users helping them in interacting with the system. This includes command-line client applications and the main executable, allowing the users/clients to perform actions such as login, upload, download, and file management.

* Application Layer:

This layer contains the core logic and processing modules of the system, handling the crucial tasks such as user authentication, session management, file upload and download handling, quota management, security checks, and secure file sharing. These modules work together and make a functional and reliable cloud storage environment.

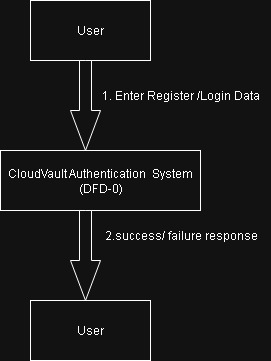
* Storage Layer:

This layer manages the actual storage of user data. It organizes files structured directories for inbox, trash, and other categorized folders, ensuring efficient file cycle management, consisting of storage, deletion, restoration. This layer abstracts away the physical storage details, provide secure and reliable data handling.

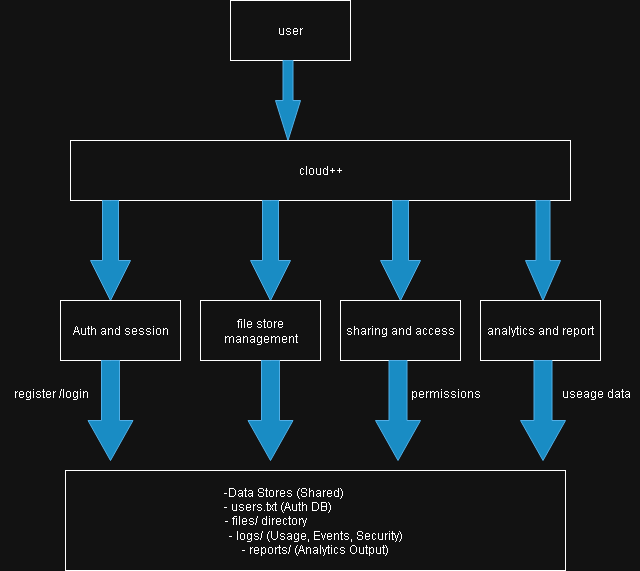
* Storage Management:

Storage Management consists of the other modules and background processes that manage and monitor storage activities. These tasks include file listing, handling uploads and downloads, maintaining logs of operations, and enforcing user quotas.

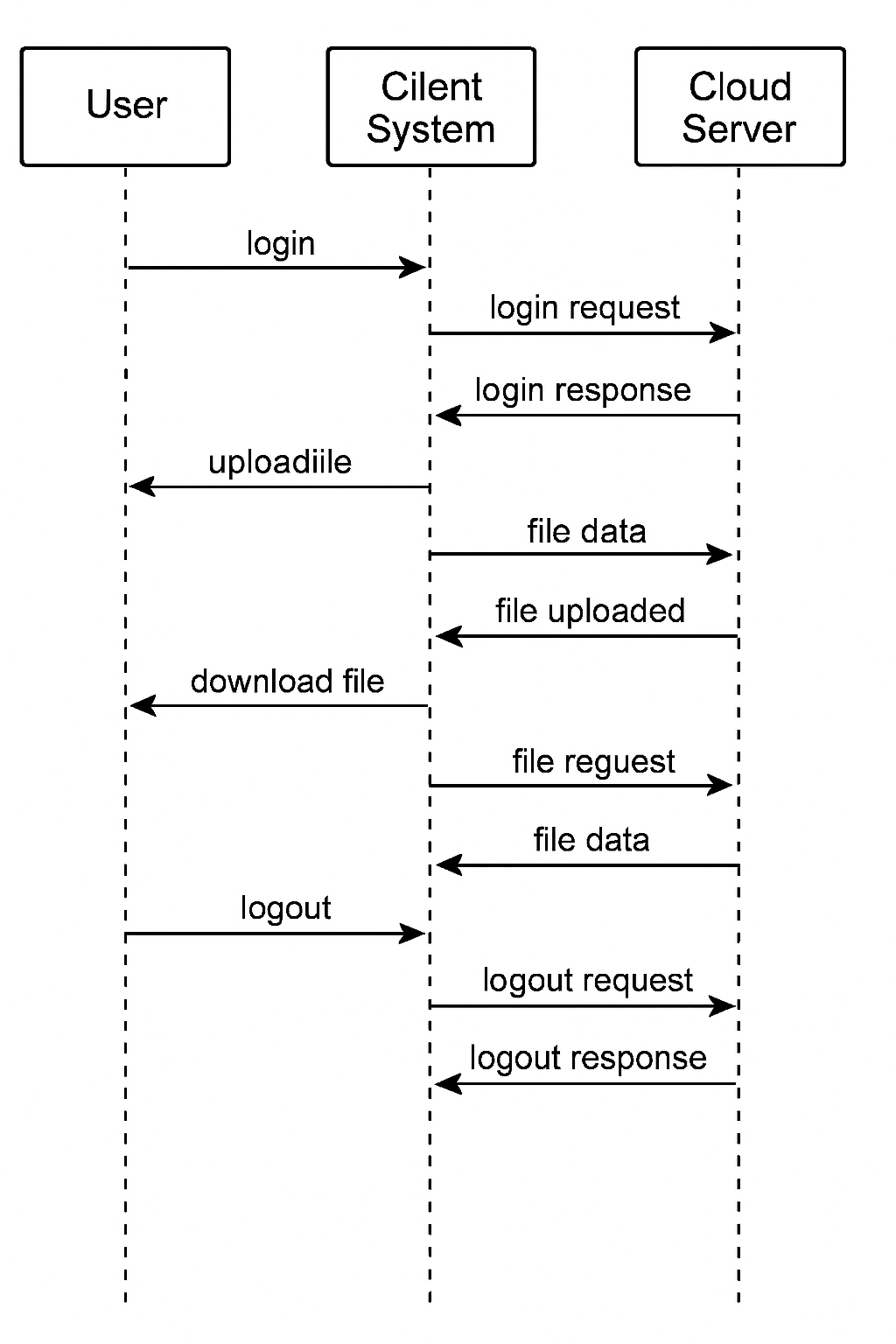
* 1. **Data Flow Diagram**
     + 1. **Level 0 Design Diagram**

****

* + - 1. **Level 1 Design Diagram**



* 1. **Sequence Diagram**



* 1. **Use Case Diagram**

**A diagram of a server

AI-generated content may be incorrect.**

* 1. **User Interface**
* Desktop or Windows with internet connection
* CLI (Client Layer Interface)
  1. **Performance**

The Cloud ++ is designed with performance and usability as their key goals, employing multi-threaded file transfers and efficient buffering techniques within modules like TransferThreader, UploadHandler, and DownloadHandler to ensure responsive and high-throughput file operations.

* 1. **Security**

This aspect is enforced rigorously through robust authentication and session management in components such as AuthClient, LoginHandler, and SessionManager, along with the permission validation and secure sharing implanted by SecureSharing and SecurityCheck.

* 1. **Reliability**

This is achieved via comprehensive error handling, fault tolerance methods like retries, and data protection through soft deletion mechanisms managed by DeleteHandle and RestoreHandler, coupled with UploadLogger.

* 1. **Maintainability**

Cloud++’s maintainability is supported by a modular design respectively under MemberA, MemberB, MemberC, and MemberD, well-documented APIs, and a build environment.

1. **Module Descriptions**
   1. **Client Layer:**

This layer handles the user interaction and request initiation, including CLI and client-side file operations.

* main.cpp – Client entry point.
* client\_storage – Local user file directories.
* AuthClient – Authentication requests from client.
* ClientUpload, ClientDownload – Manage the client-side file uploads/downloads.
* connection\_manager, login\_handler – connection and login input.
  1. **Server Layer:**

This includes the core logic including authentication, session management, file operations, quota management and security.

* AuthHandler, RegisterHandler, SessionManager, UserValidator, Password Manager – User Authentication and Session Management.
* UploadHandler, DownloadHandler, TransferThreader, QuotaManager, Upload Logger, FileLister – File operations on the server side.
* Delete Handler, PermanentDeleteHandler, RestoreHandler, Inbox, SecureSharing, SecurityCheck, TrashViewer – File Lifecycle and security.
* ConnectionManager, NetworkUtils – Networking and connection.
  1. **Storage Layer**

This layer manages the physical storage of user data in organized folders based on the user and category, with inbox and trash states, and persistent files for quotas and logs.

* storage/ - Contains active and deleted files in respective their directories.
* Trash/ - Contains files that are marked for permanent deletion.
  1. **Admin and Utility Layer**

This provides administrative tools, system monitoring, cleanup and analytics.

* AdminPanel/ - Includes the admin panel.
* CleanupTool/, LogReader/ - These are maintenance tools.
* StorageMonitor/ - Monitoring.
* UserInsights/ - Analytics.

1. **Environment Description**
   1. **Hardware Requirements**

**Server**

* RAM: Minimum 4GB
* Storage: 500mb

**Client**

* RAM: Minimum 2GB
* Storage: Based on the transferred files.
  1. **Software Requirements**

**Server**

* OS: Windows 10/11 x64
* Compiler: C++17 compatible, MSVC
* Libraries: Standard C++ libs, WinSock2

**Client:**

* OS: Windows 10/11
* Runtime: Compatible C++ runtime
* Interface: Command-line terminal

1. **Error Handling**

* Should and errors be encountered a notification will be displayed as to what has gone wrong.
* The server will check for the user if they are registered or not in the data base.
* If there is any interruption while uploading the user must check the connection and reupload it.

1. **Introduction**

**LOW LEVEL DESIGN**

* 1. **Description**

The Cloud++ cloud storage system is designed to offer users a reliable and secure cloud environment for the users to transfer files. It incorporates user authentication, session tracking, uploading, downloading, deletion and restoration. All these through a layer modular architecture. It also ensures the clear separation between client interaction, data storage and ease of maintenance while also facilitating quota management and secure client-to-client file sharing.

* 1. **Purpose**

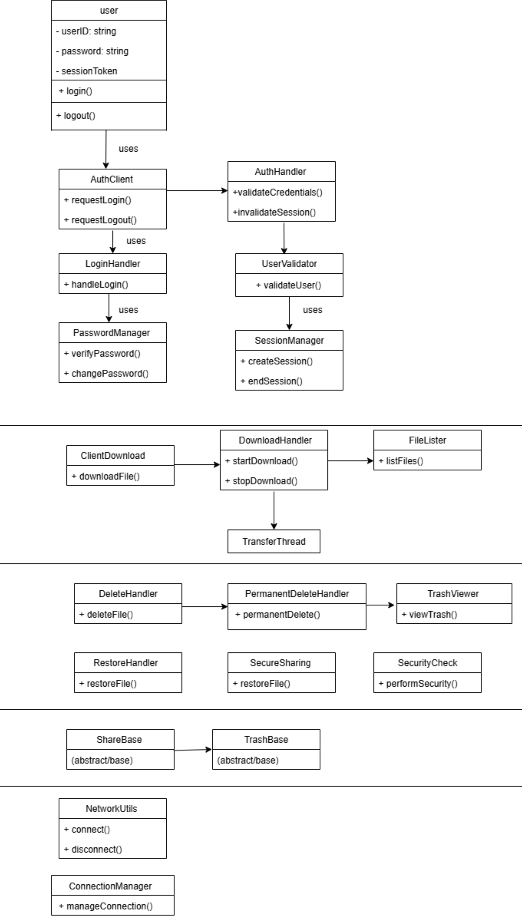
The purpose of Cloud ++ is to deliver a secure and reliable platform that enables users to remotely store, access, share and manage their files with ease. Client also has the accessibility to delete or restore a file.

* 1. **References**
* [**Beej’s Guide to Networking**](https://www.bing.com/search?pglt=163&q=beejs+guide+to+socket+programming&cvid=8d7a4830eb354f67bdabcd7c7dbc89e1&gs_lcrp=EgRlZGdlKgYIABBFGDkyBggAEEUYOTIGCAEQABhAMgYIAhAAGEAyCAgDEOkHGPxV0gEINzAwMmowajGoAgCwAgA&FORM=ANNAB1&PC=U531)
* [**Socket Programming in C++**](https://www.tutorialspoint.com/cplusplus/cpp_socket_programming.htm)

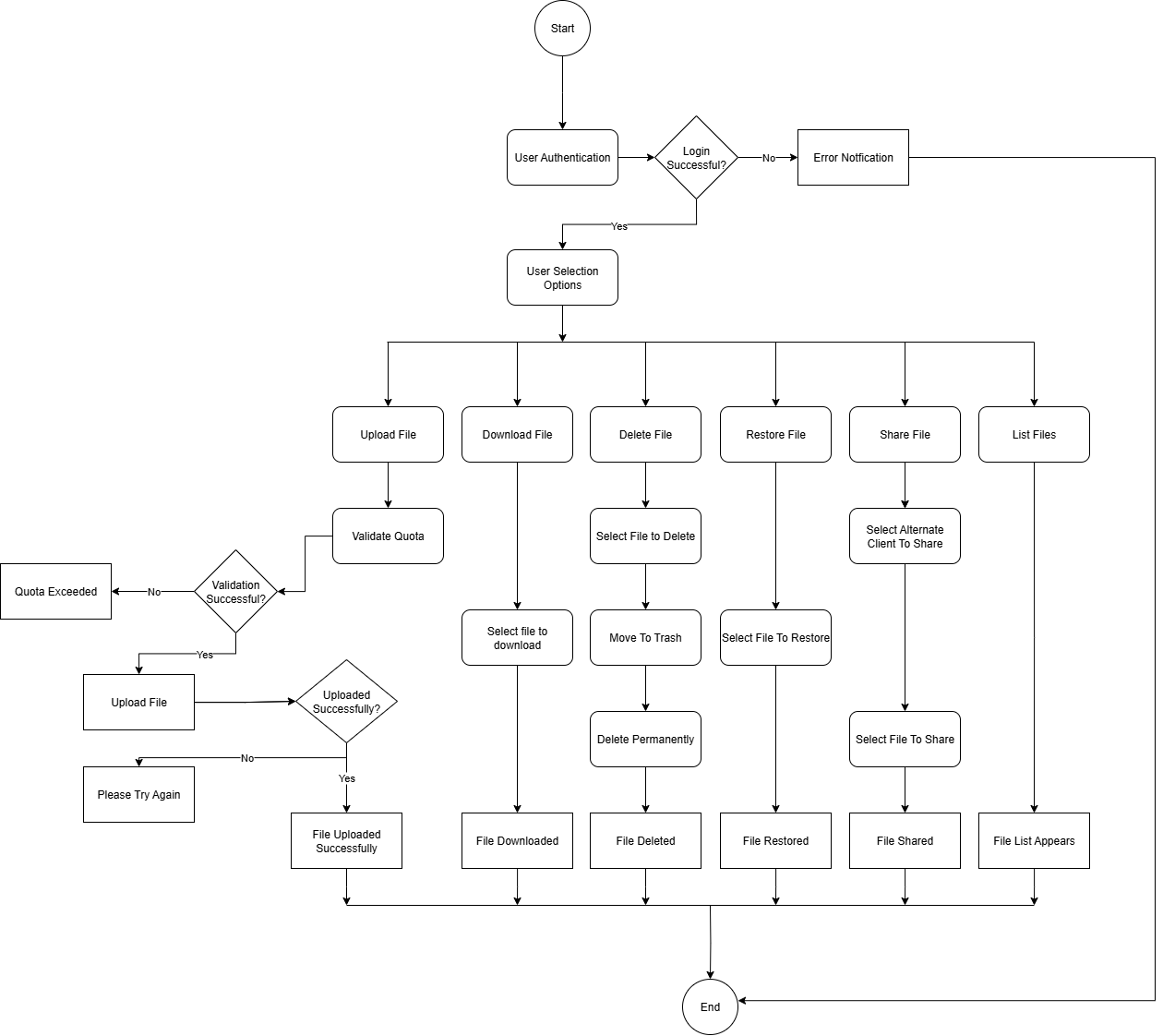
1. **Detailed System Design**
   1. **Design Descriptions**

The Cloud++ cloud storage system utilizes a well-layered, modular architecture with a reliable and secure file management. Here, the users (clients) interact via a CLI, initiating actions such as file uploading, downloading, deleting and sharing. Requests are transmitted over a TCP/IP to the server, which performs credential validation, manages sessions, and dispatch the respective operations to specialized handlers. Uploads and downloads leverage transfers with quota management and logging for auditability. Files states transition between active and trash partitions within organized storage directories, supporting file lifecycle management and secure sharing client-to-client sharing.

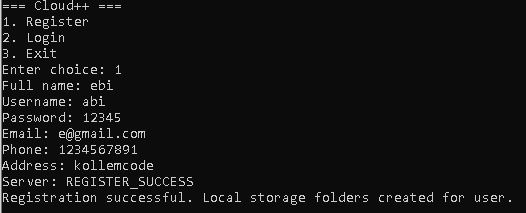
* 1. **Class Diagram**

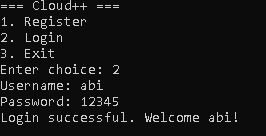
****

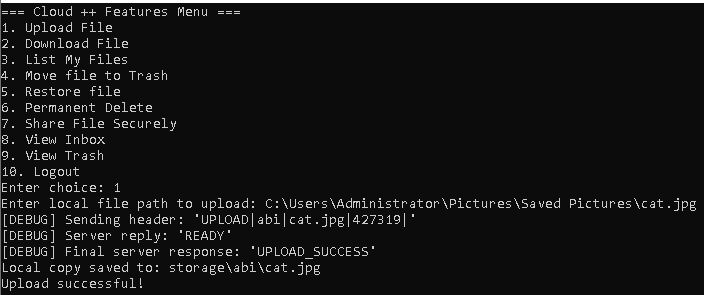
* 1. **Activity Diagram**

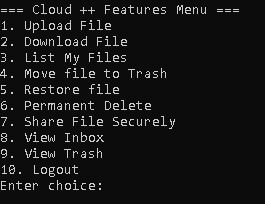


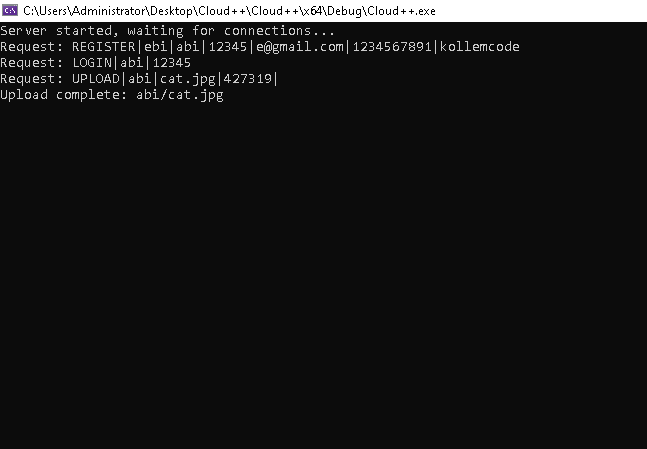
1. **Demo**

****

****

****

****

****