

# Secure File Storage System

## 1. Title Page

Project Name: Secure File Storage System

Group Members: Usman Ali

AbuBakr Butt

Maisam Raza

## 2. Abstract

This project, Secure File Storage System, addresses the growing need for secure and efficient file storage. It combines encryption techniques with dynamic file management capabilities to ensure data confidentiality and usability. The project employs modular design principles and integrates data structures and algorithms (DSA) for efficient file operations. This document outlines the design, implementation, testing, and usage of the system.

## 3. Introduction

### Purpose:

The Secure File Storage System is designed to:

- Safeguard sensitive data through encryption.
- Enable dynamic file operations such as creation, deletion, and searching.
- Provide user authentication to prevent unauthorized access.

### Importance:

In an era of increasing data breaches, this system ensures that sensitive information remains confidential and accessible only to authorized users. By integrating file management with encryption, the project offers a comprehensive solution for secure storage.

## 4. System Requirements

### Hardware Requirements:

Processor: Intel Core i3 or higher

RAM: 4 GB minimum

Storage: 1 GB free disk space

### Software Requirements:

Operating System: Windows 10 or higher

Compiler: GCC (MinGW for Windows)

IDE: Visual Studio Code

#### Dependencies:

Standard Template Library (STL) for data structures  
C++17 or higher

## 5. System Design

#### Architecture Diagram:

The system architecture consists of three layers:

1. Input Layer: Handles user commands and interactions.
2. Processing Layer: Implements encryption, file management, and authentication.
3. Storage Layer: Stores encrypted files securely.

#### Flowchart:

1. User logs in using valid credentials.
2. System authenticates the user.
3. User performs file operations (e.g., create, delete, search).
4. Files are encrypted/decrypted during storage/retrieval.

#### UML Diagram:

Include a class diagram showcasing the relationships between key classes, such as EncryptFile, DeleteFile, Login, etc.

## 6. Implementation Details

#### Coding Techniques:

- Modular Design: Functions are organized into separate `.cpp` and `.h` files.
- Dynamic Memory Management: Used linked lists for file tracking.
- Encryption: Simple XOR-based encryption for file security.

#### Frameworks and Libraries:

Standard Template Library (STL) for data structures like lists.  
File handling libraries for disk I/O operations.

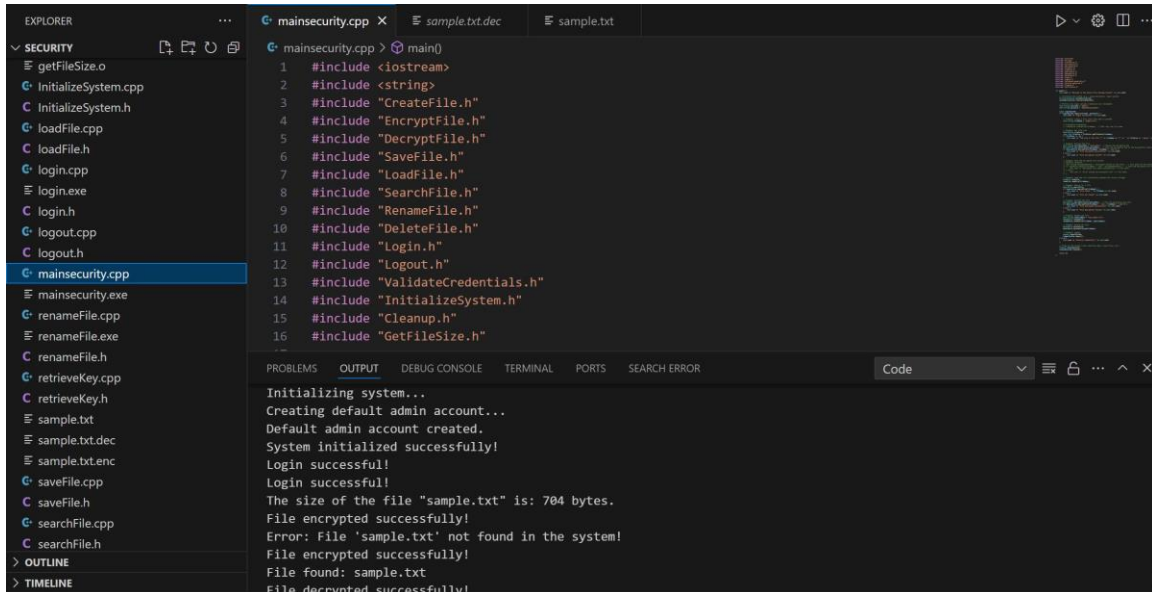
## 7. Testing and Results

#### Test Cases:

1. Authentication:
  - Input: Valid username and password.
  - Expected Result: User logged in successfully.
  - Outcome: Passed
2. File Deletion:
  - Input: Delete an existing file.
  - Expected Result: File removed from disk and file list.
  - Outcome: Passed

## Screenshots:

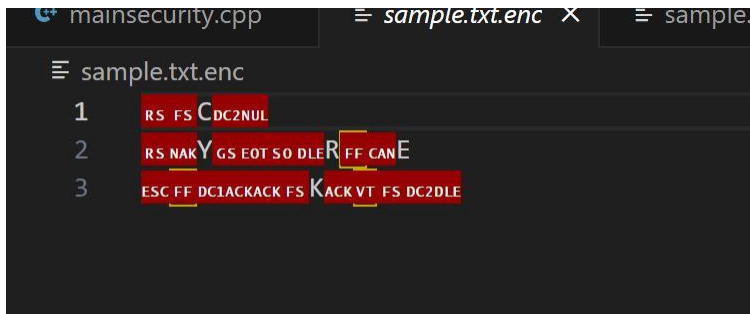
### CODE RUN SUCCESFUL



```
mainsecurity.cpp main()
1 #include <iostream>
2 #include <string>
3 #include "CreateFile.h"
4 #include "EncryptFile.h"
5 #include "DecryptFile.h"
6 #include "SaveFile.h"
7 #include "LoadFile.h"
8 #include "SearchFile.h"
9 #include "RenameFile.h"
10 #include "DeleteFile.h"
11 #include "Login.h"
12 #include "Logout.h"
13 #include "ValidateCredentials.h"
14 #include "InitializeSystem.h"
15 #include "Cleanup.h"
16 #include "GetFileSize.h"

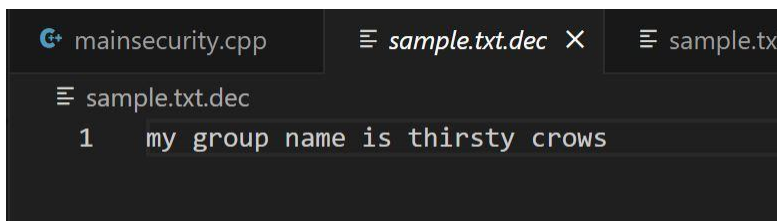
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR Code
Initializing system...
Creating default admin account...
Default admin account created.
System initialized successfully!
Login successful!
Login successful!
The size of the file "sample.txt" is: 704 bytes.
File encrypted successfully!
Error: File 'sample.txt' not found in the system!
File encrypted successfully!
File found: sample.txt
File decrypted successfully!
```

### ENCRYPTED TEXT



```
mainsecurity.cpp sample.txt.enc sample.txt
sample.txt.enc
1 RS FS CDC2NUL
2 RS NAKY GS EOT SO DLE R FF CAN E
3 ESC FF DCIACKACK FS KACK VT FS DC2DLE
```

### DECRYPTED TEXT



```
mainsecurity.cpp sample.txt.dec sample.txt
sample.txt.dec
1 my group name is thirsty crows
```

## Results:

Successfully implemented and tested all core functionalities.

## 8. User Guide

### Installation:

1. Install GCC compiler and Visual Studio Code.
2. Clone the project repository.

### Configuration:

1. Ensure all `.cpp`` and `.h`` files are in the same directory.
2. Use the provided `tasks.json`` to configure build tasks in VS Code.

### Running the Project:

1. Open the terminal in the project folder.
2. Run `mainsecurity.exe`` to start the system.
3. Follow the on-screen instructions for file operations.

## 9. Conclusion

The Secure File Storage System successfully addresses the need for a secure and dynamic file management solution. By integrating encryption, authentication, and efficient file operations, the project ensures data confidentiality and usability. Future enhancements could include advanced encryption standards and multi-user support.

## 10. References

- C++ Standard Library Documentation
- GCC Compiler Documentation
- [External Tutorials or Articles Used]