GOOGLE DRIVE SIMULATOR

Data Structures and Algorithms Lab.

Project Report

Console based Google Drive Simulator

Developed by:

- Muhammad Hamza

Introduction

The objective of this project is to develop a console-based Google Drive-like file management system in **C++**. This system allows users to register, log in, create, delete, and manage files and folder. It also includes functionalities like a recycle bin**,** file sharing, and user authentication**.**

Tools and Technologies

- **Programming language:** C++
- **IDE:** Visual Studio
- **File I/O:** <fstream>
- **Libraries used:** <iostream>, <string>, <ctime>, <sstream>

System features

- **User Authentication:** Secure authentication system using password and security question.
- **File/Folder management:** Create, delete, navigate and display folder and files.
- **Recycle Bin:** Recover and Display deleted files.
- **File sharing:** Share files between registered users.
- **Password recovery:** Reset password using security question.

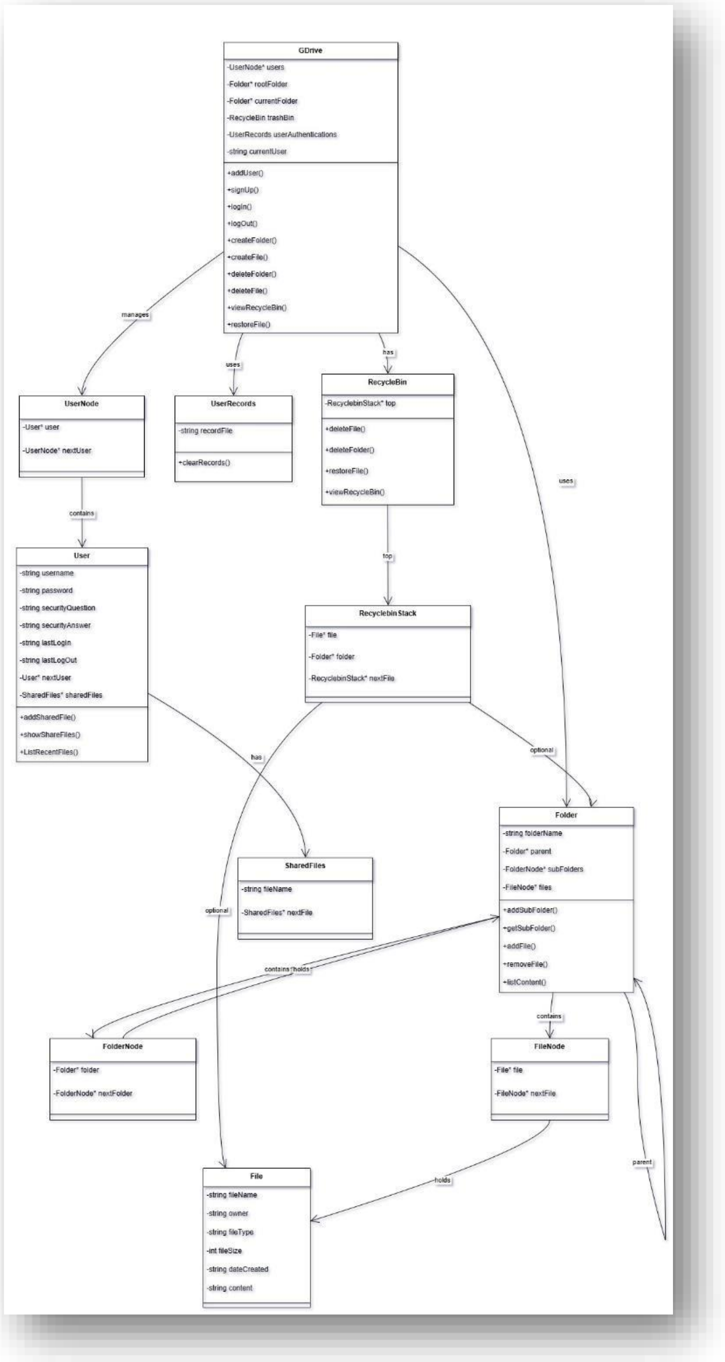
Data Structures Used

- **Linked List:** Manage user records and file/folder nodes.
- **Stack:** Recycle Bin implementation
- **Queue:** Recent activity
- **Tree:** Folder hierarchy
- **Graph:** File sharing connection
- **File I/O:** Session tracking

System Architecture

Class structure overview

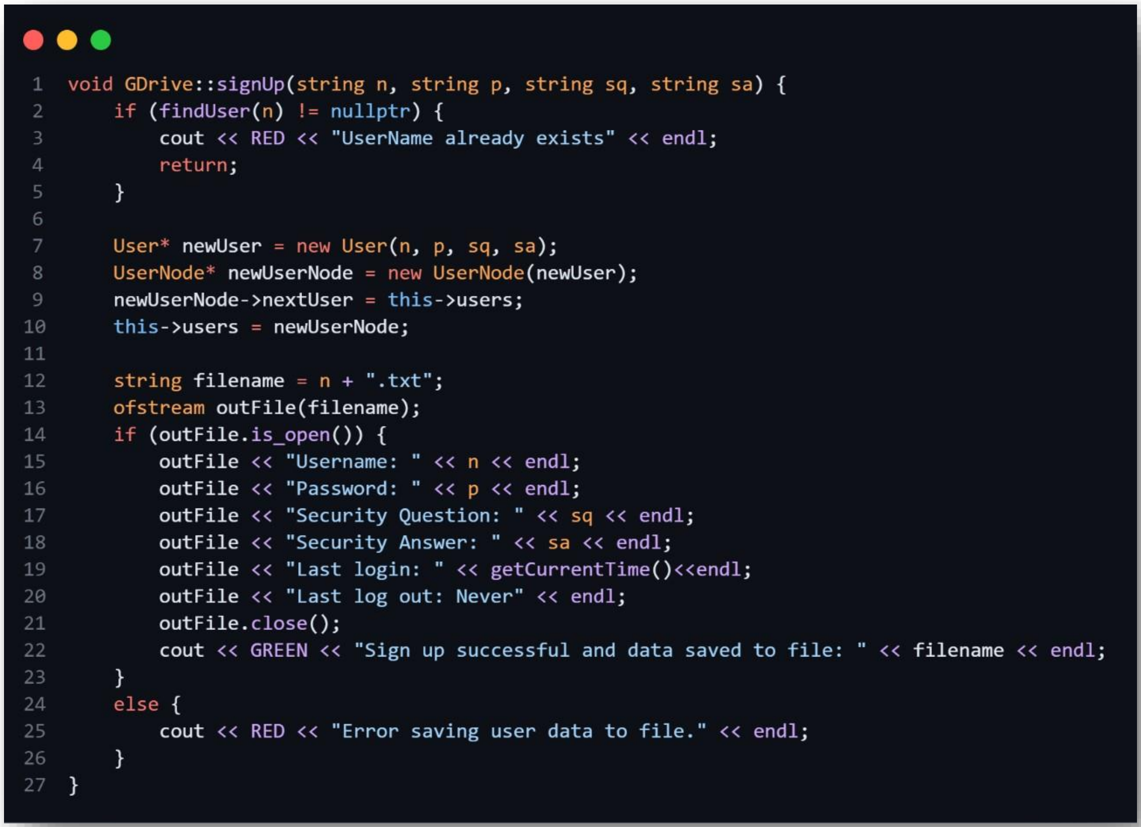
- **User:** Stores user credentials and shared files.
- **Folder and FolderNode:** Represents hieratical folder structure.
- **File and FileNode:** Represents files and their metadata.
- **RecycleBin:** Recycle bin implementation.
- **GDrive:** Main controller managing authentication and actions.



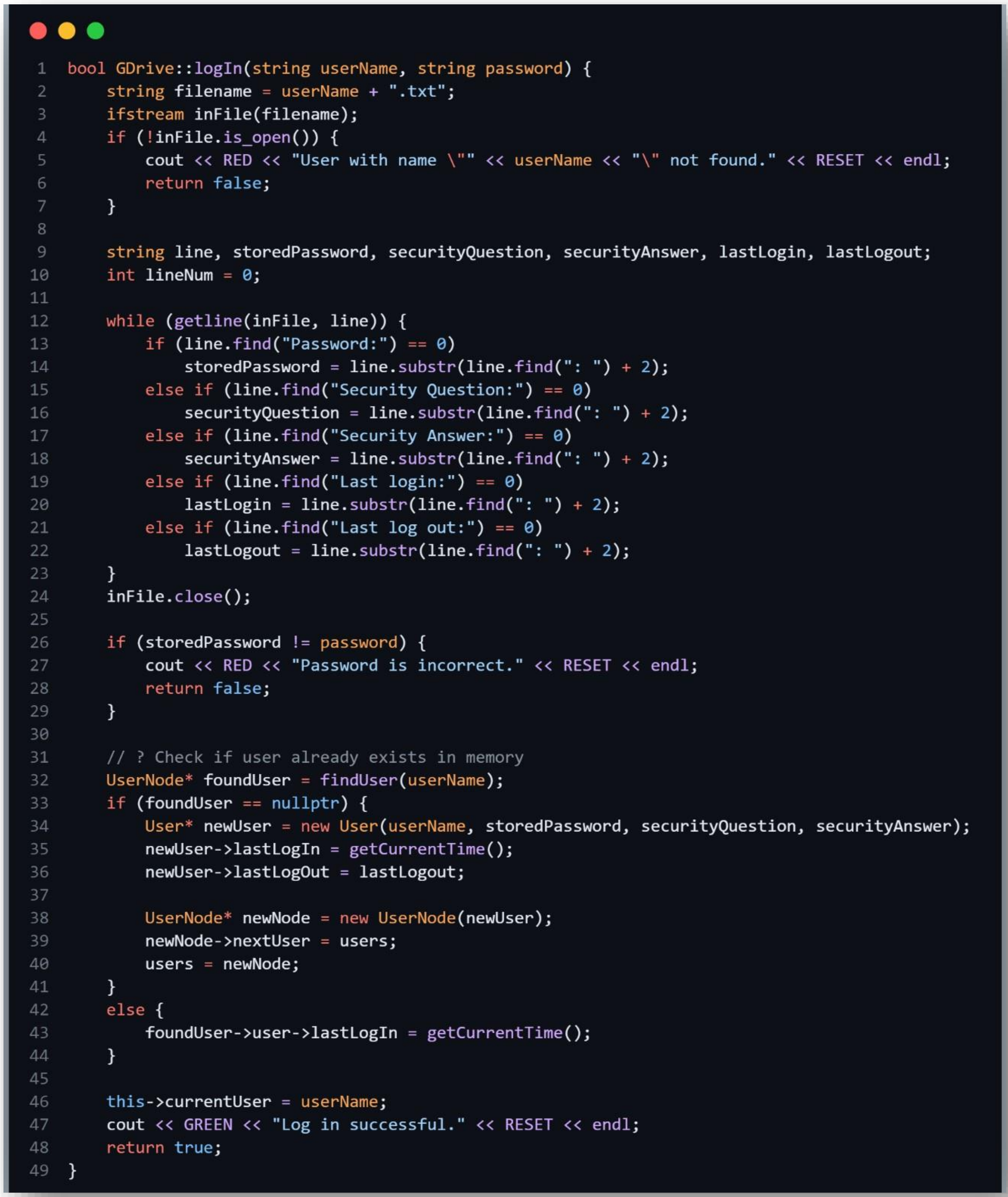
Core functional modules

User Authentication

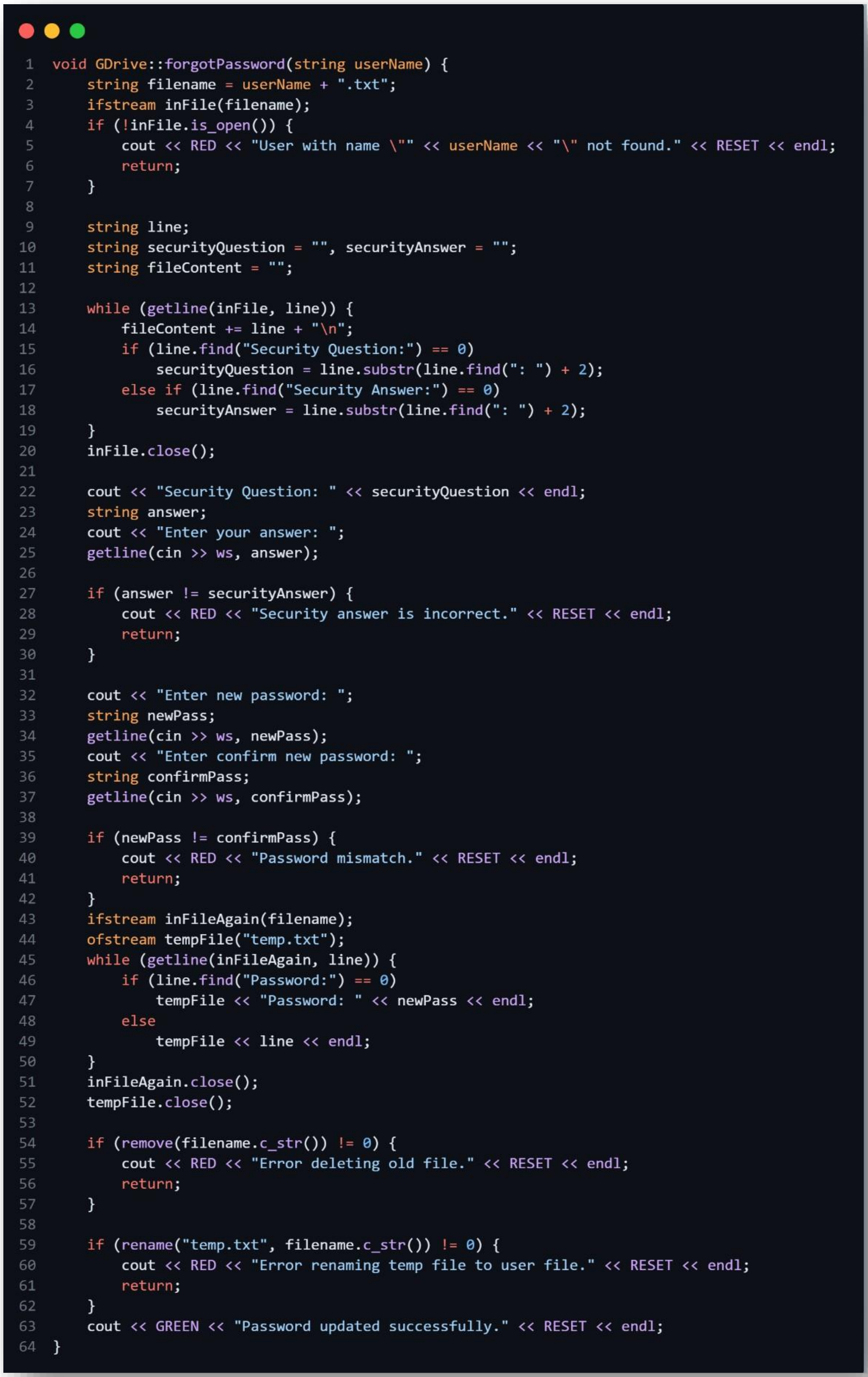
- Sign Up/Log in with security question and answer.
- Password recovery and file based storage for user metadata.



```cpp
void GDrive::signUp(string n, string p, string sq, string sa) {
    if (findUser(n) != nullptr) {
        cout << RED << "UserName already exists" << endl;
        return;
    }

    User* newUser = new User(n, p, sq, sa);
    UserNode* newUserNode = new UserNode(newUser);
    newUserNode->nextUser = this->users;
    this->users = newUserNode;

    string filename = n + ".txt";
    ofstream outFile(filename);
    if (outFile.is_open()) {
        outFile << "Username: " << n << endl;
        outFile << "Password: " << p << endl;
        outFile << "Security Question: " << sq << endl;
        outFile << "Security Answer: " << sa << endl;
        outFile << "Last login: " << getCurrentTime()<<endl;
        outFile << "Last log out: Never" << endl;
        outFile.close();
        cout << GREEN << "Sign up successful and data saved to file: " << filename << endl;
    }
    else {
        cout << RED << "Error saving user data to file." << endl;
    }
}
```



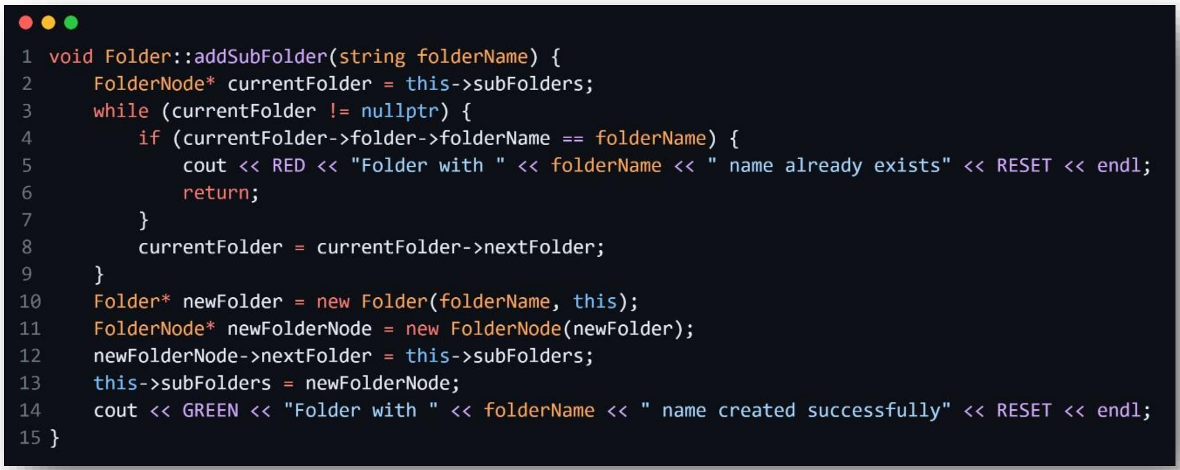
```cpp
bool GDrive::logIn(string userName, string password) {
    string filename = userName + ".txt";
    ifstream inFile(filename);
    if (!inFile.is_open()) {
        cout << RED << "User with name \"" << userName << "\" not found." << RESET << endl;
        return false;
    }

    string line, storedPassword, securityQuestion, securityAnswer, lastLogin, lastLogout;
    int lineNum = 0;

    while (getline(inFile, line)) {
        if (line.find("Password:") == 0)
            storedPassword = line.substr(line.find(": ") + 2);
        else if (line.find("Security Question:") == 0)
            securityQuestion = line.substr(line.find(": ") + 2);
        else if (line.find("Security Answer:") == 0)
            securityAnswer = line.substr(line.find(": ") + 2);
        else if (line.find("Last login:") == 0)
            lastLogin = line.substr(line.find(": ") + 2);
        else if (line.find("Last log out:") == 0)
            lastLogout = line.substr(line.find(": ") + 2);
    }
    inFile.close();

    if (storedPassword != password) {
        cout << RED << "Password is incorrect." << RESET << endl;
        return false;
    }

    // ? Check if user already exists in memory
    UserNode* foundUser = findUser(userName);
    if (foundUser == nullptr) {
        User* newUser = new User(userName, storedPassword, securityQuestion, securityAnswer);
        newUser->lastLogIn = getCurrentTime();
        newUser->lastLogOut = lastLogout;

        UserNode* newNode = new UserNode(newUser);
        newNode->nextUser = users;
        users = newNode;
    }
    else {
        foundUser->user->lastLogIn = getCurrentTime();
    }

    this->currentUser = userName;
    cout << GREEN << "Log in successful." << RESET << endl;
    return true;
}
```



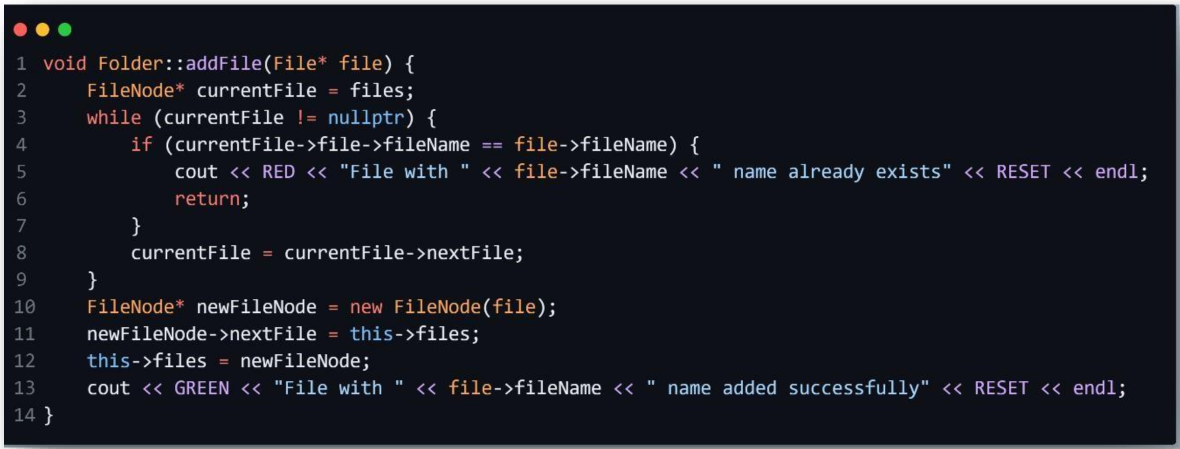
```cpp
void GDrive::forgotPassword(string userName) {
    string filename = userName + ".txt";
    ifstream inFile(filename);
    if (!inFile.is_open()) {
        cout << RED << "User with name \"" << userName << "\" not found." << RESET << endl;
        return;
    }

    string line;
    string securityQuestion = "", securityAnswer = "";
    string fileContent = "";

    while (getline(inFile, line)) {
        fileContent += line + "\n";
        if (line.find("Security Question:") == 0)
            securityQuestion = line.substr(line.find(": ") + 2);
        else if (line.find("Security Answer:") == 0)
            securityAnswer = line.substr(line.find(": ") + 2);
    }
    inFile.close();

    cout << "Security Question: " << securityQuestion << endl;
    string answer;
    cout << "Enter your answer: ";
    getline(cin >> ws, answer);

    if (answer != securityAnswer) {
        cout << RED << "Security answer is incorrect." << RESET << endl;
        return;
    }

    cout << "Enter new password: ";
    string newPass;
    getline(cin >> ws, newPass);
    cout << "Enter confirm new password: ";
    string confirmPass;
    getline(cin >> ws, confirmPass);

    if (newPass != confirmPass) {
        cout << RED << "Password mismatch." << RESET << endl;
        return;
    }
    ifstream inFileAgain(filename);
    ofstream tempFile("temp.txt");
    while (getline(inFileAgain, line)) {
        if (line.find("Password:") == 0)
            tempFile << "Password: " << newPass << endl;
        else
            tempFile << line << endl;
    }
    inFileAgain.close();
    tempFile.close();

    if (remove(filename.c_str()) != 0) {
        cout << RED << "Error deleting old file." << RESET << endl;
        return;
    }

    if (rename("temp.txt", filename.c_str()) != 0) {
        cout << RED << "Error renaming temp file to user file." << RESET << endl;
        return;
    }
    cout << GREEN << "Password updated successfully." << RESET << endl;
}
```

File and Folder Operations

- Create/Delete folders and files inside current directory
- Traverse folders/files



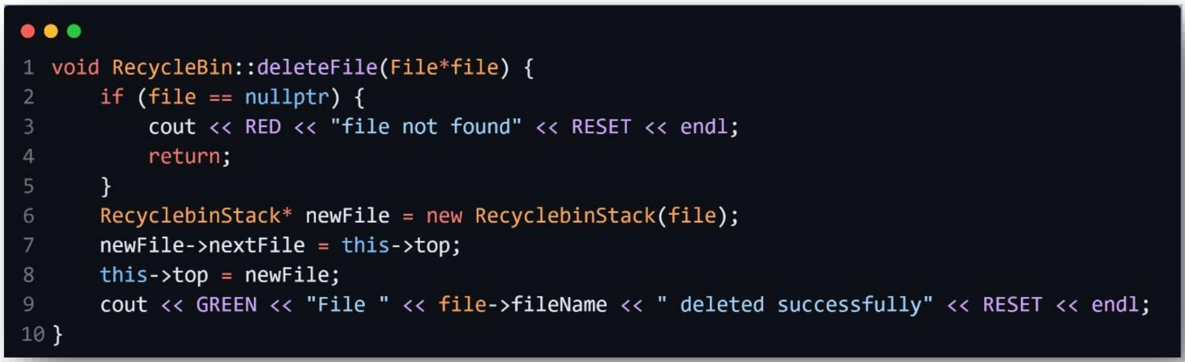
```cpp
void Folder::addSubFolder(string folderName) {
    FolderNode* currentFolder = this->subFolders;
    while (currentFolder != nullptr) {
        if (currentFolder->folder->folderName == folderName) {
            cout << RED << "Folder with " << folderName << " name already exists" << RESET << endl;
            return;
        }
        currentFolder = currentFolder->nextFolder;
    }
    Folder* newFolder = new Folder(folderName, this);
    FolderNode* newFolderNode = new FolderNode(newFolder);
    newFolderNode->nextFolder = this->subFolders;
    this->subFolders = newFolderNode;
    cout << GREEN << "Folder with " << folderName << " name created successfully" << RESET << endl;
}
```

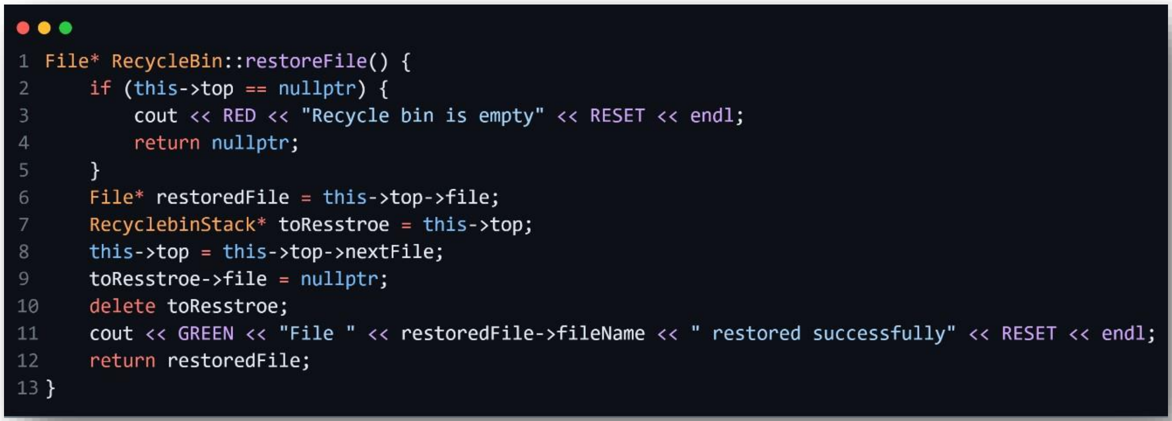


```cpp
void Folder::addFile(File* file) {
    FileNode* currentFile = files;
    while (currentFile != nullptr) {
        if (currentFile->file->fileName == file->fileName) {
            cout << RED << "File with " << file->fileName << " name already exists" << RESET << endl;
            return;
        }
        currentFile = currentFile->nextFile;
    }
    FileNode* newFileNode = new FileNode(file);
    newFileNode->nextFile = this->files;
    this->files = newFileNode;
    cout << GREEN << "File with " << file->fileName << " name added successfully" << RESET << endl;
}
```

Recycle Bin

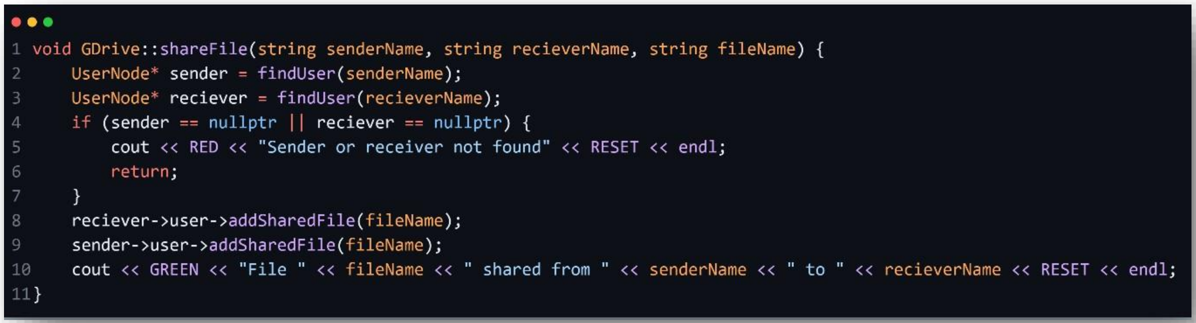
- Uses a stack to hold deleted files
- Files can be restored


```cpp
void RecycleBin::deleteFile(File*file) {
    if (file == nullptr) {
        cout << RED << "file not found" << RESET << endl;
        return;
    }
    RecyclebinStack* newFile = new RecyclebinStack(file);
    newFile->nextFile = this->top;
    this->top = newFile;
    cout << GREEN << "File " << file->fileName << " deleted successfully" << RESET << endl;
}
```

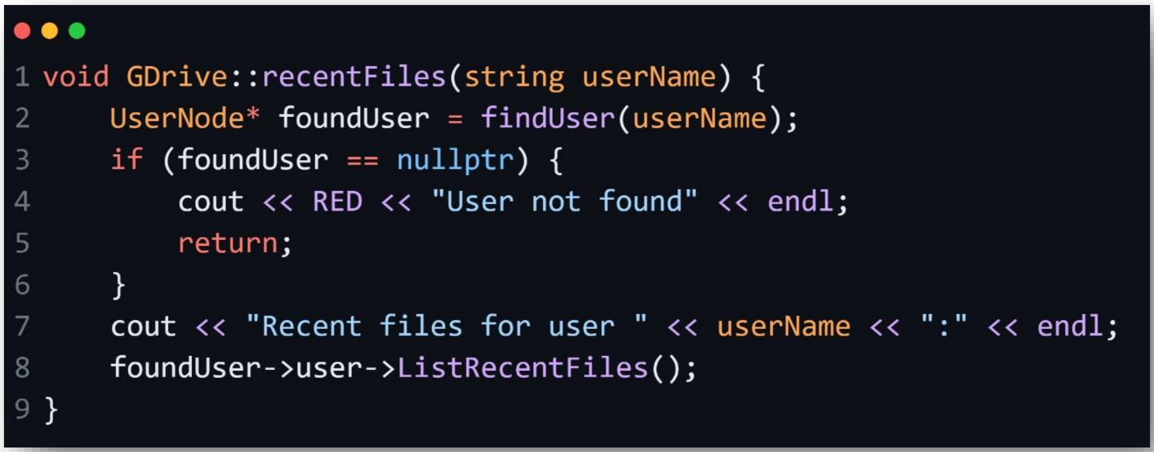

```cpp
File* RecycleBin::restoreFile() {
    if (this->top == nullptr) {
        cout << RED << "Recycle bin is empty" << RESET << endl;
        return nullptr;
    }
    File* restoredFile = this->top->file;
    RecyclebinStack* toResstroe = this->top;
    this->top = this->top->nextFile;
    toResstroe->file = nullptr;
    delete toResstroe;
    cout << GREEN << "File " << restoredFile->fileName << " restored successfully" << RESET << endl;
    return restoredFile;
}
```

File Sharing and Recent Files

- Files can be shared across users
- Recent files access is tracked via Queue


```cpp
void GDrive::shareFile(string senderName, string recieverName, string fileName) {
    UserNode* sender = findUser(senderName);
    UserNode* reciever = findUser(recieverName);
    if (sender == nullptr || reciever == nullptr) {
        cout << RED << "Sender or receiver not found" << RESET << endl;
        return;
    }
    reciever->user->addSharedFile(fileName);
    sender->user->addSharedFile(fileName);
    cout << GREEN << "File " << fileName << " shared from " << senderName << " to " << recieverName << RESET << endl;
}
```

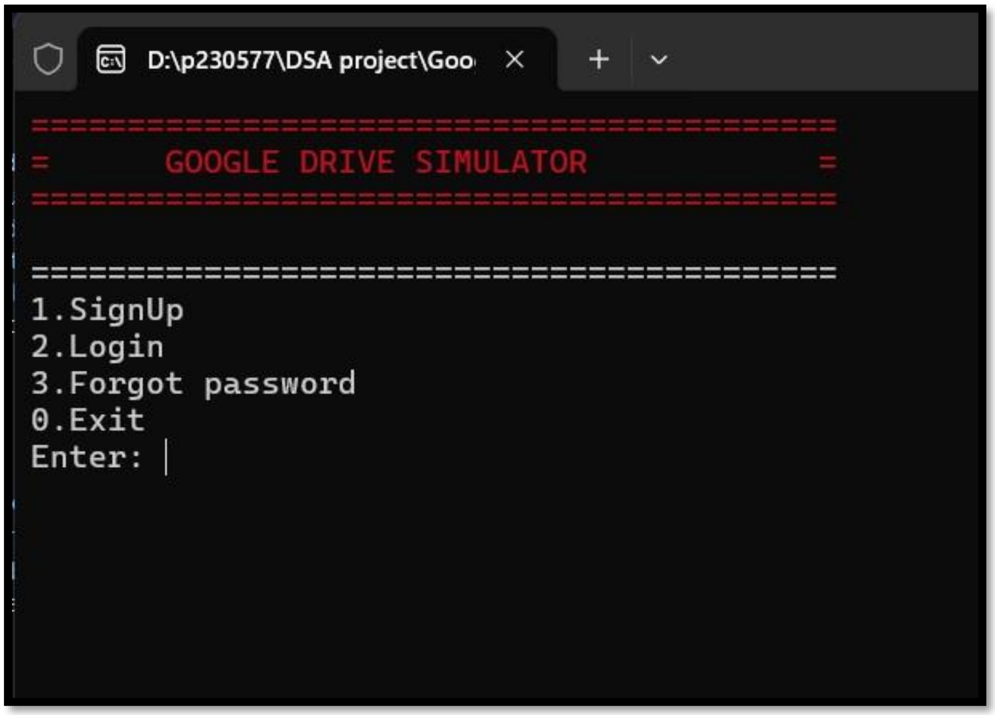
-



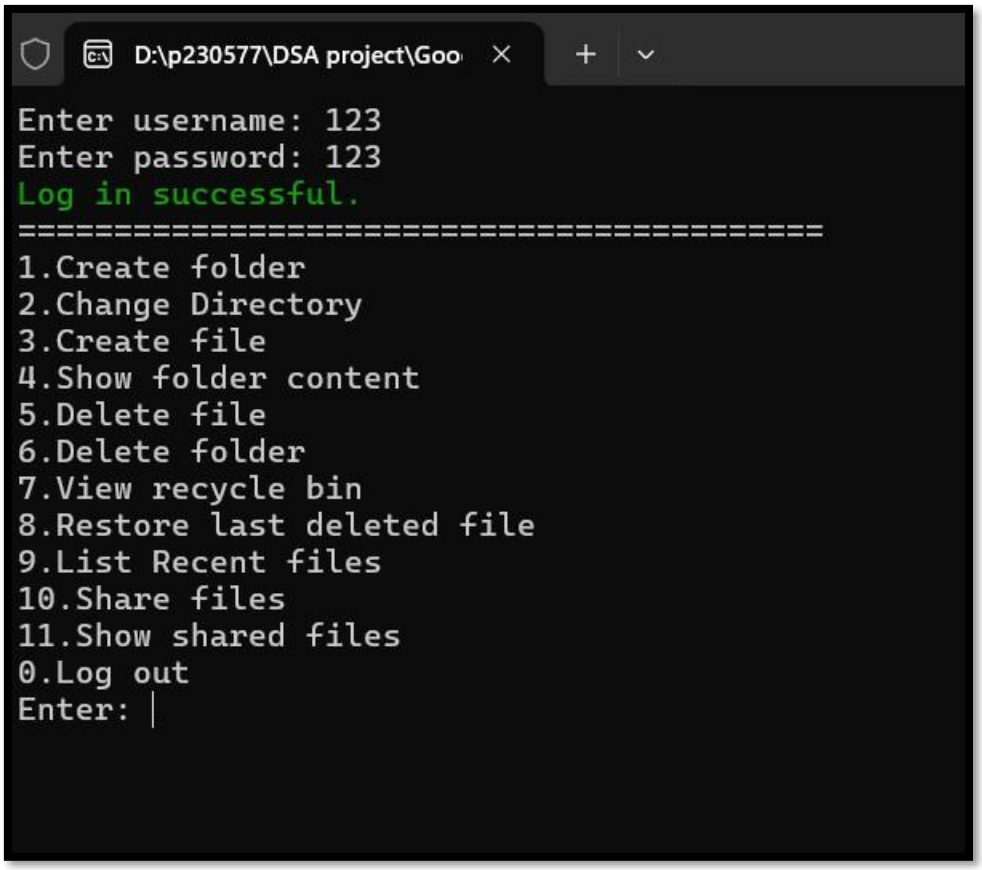
```cpp
void GDrive::recentFiles(string userName) {
    UserNode* foundUser = findUser(userName);
    if (foundUser == nullptr) {
        cout << RED << "User not found" << endl;
        return;
    }
    cout << "Recent files for user " << userName << ":" << endl;
    foundUser->user->ListRecentFiles();
}
```

Sample Execution Flow

- Start the program
- Sign up a new user
- Login
- Create folder/file
- Share files with another user
- Delete and restore from recycle bin
- Logout



```
==========================================
=      GOOGLE DRIVE SIMULATOR            =
==========================================

==========================================
1.SignUp
2.Login
3.Forgot password
0.Exit
Enter:
```



```
Enter username: 123
Enter password: 123
Log in successful.
==========================================
1.Create folder
2.Change Directory
3.Create file
4.Show folder content
5.Delete file
6.Delete folder
7.View recycle bin
8.Restore last deleted file
9.List Recent files
10.Share files
11.Show shared files
0.Log out
Enter:
```

Testing and Validation

The system was tested with multiple users and file operations. Validation covered:

- Incorrect password/username scenario
- Deletion of non-existing file
- Recovery of multiple file types
- Folder navigation

All features function correctly (except deletion and file/folder existence) and memory was managed by destructors and proper cleanup.

Conclusion

The project successfully demonstrates how fundamental data structures can be combined to simulate a google drive. It reinforces core software design principles such as:

- Object oriented programming
- File handling
- Memory management
- Modularization

The project can be further extended by adding:

- GUI
- Cloud sync using APIs
- Role based access control