5/5/2024



Name: Adnan & Abdullah zain

Roll No. 21F-9220 & 22f-3079

Section: Bcs-4b

Instructor: mam.rabia anwer



LAB project# final

DATA LAB

**Question 1:**

**Code:**

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <windows.h>

using namespace std;

class File {

public:

string name;

string data;

File\* next;

File(const string& n, const string& d) : name(n), data(d), next(nullptr)

{}

};

class Comment {

public:

string comment;

Comment\* next;

Comment(const string& c) : comment(c), next(nullptr)

{}

};

class Repository {

public:

string name;

File\* files;

Comment\* comments;

bool visibility;

Repository\* next;

Repository(const string& n) : name(n), files(nullptr), comments(nullptr), next(nullptr)

{}

void add\_File(const string& fileName, const string& fileData)

{

File\* newFile = new File(fileName, fileData);

if (files == nullptr)

files = newFile;

else {

File\* temp = files;

while (temp->next != nullptr)

temp = temp->next;

temp->next = newFile;

}

}

void add\_Comment(const string& commentText)

{

Comment\* newComment = new Comment(commentText);

if (comments == nullptr)

comments = newComment;

else {

Comment\* temp = comments;

while (temp->next != nullptr)

temp = temp->next;

temp->next = newComment;

}

}

void displayVisibility() {

cout << "Visibility of repository '" << name << "': " << (visibility ? "Public" : "Private") << endl;

}

void setVisibility(bool visibility)

{

this->visibility = visibility;

}

};

// Forward declaration of User

class User;

// Node structure for the graph (linked list representation)

struct UserNode {

User\* user;

UserNode\* next;

UserNode(User\* u) : user(u), next(nullptr)

{}

};

// Class representing a User

class User {

public:

string username;

string password;

int followers;

Repository\* repositories;

UserNode\* followingHead; // Head pointer for the list of users being followed

User\* next;

User(const string& u, const string& p) : username(u), password(p), followers(0), repositories(nullptr), followingHead(nullptr), next(nullptr) {}

// Function to follow another user

void follow(User& otherUser)

{

if (&otherUser == this)

{

cout << "You cannot follow yourself." << endl;

return;

}

// Check if already following

UserNode\* curr = followingHead;

while (curr != nullptr)

{

if (curr->user == &otherUser)

{

cout << "You are already following " << otherUser.username << endl;

return;

}

curr = curr->next;

}

// Add the user to the following list

UserNode\* newNode = new UserNode(&otherUser);

newNode->next = followingHead;

followingHead = newNode;

// Increment other user's followers count

otherUser.followers++;

cout << "You are now following " << otherUser.username << endl;

}

// Function to unfollow another user

void unfollow(User& otherUser)

{

if (&otherUser == this)

{

cout << "You cannot unfollow yourself." << endl;

return;

}

UserNode\* prev = nullptr;

UserNode\* curr = followingHead;

// Search for the user to unfollow

while (curr != nullptr && curr->user != &otherUser)

{

prev = curr;

curr = curr->next;

}

// If found, remove from the following list

if (curr != nullptr)

{

if (prev != nullptr)

{

prev->next = curr->next;

}

else {

followingHead = curr->next;

}

delete curr;

// Decrement other user's followers count

otherUser.followers--;

cout << "You have unfollowed " << otherUser.username << endl;

}

else

{

cout << "You are not following " << otherUser.username << endl;

}

}

// Function to display the users being followed

void display\_Following()

{

cout << "Users you are following:" << endl;

UserNode\* curr = followingHead;

while (curr != nullptr)

{

cout << curr->user->username << endl;

curr = curr->next;

}

}

void add\_Repository(const string& repoName, bool visibility)

{

Repository\* newRepo = new Repository(repoName);

newRepo->visibility = visibility;

if (repositories == nullptr)

repositories = newRepo;

else

{

Repository\* temp = repositories;

while (temp->next != nullptr)

temp = temp->next;

temp->next = newRepo;

}

}

Repository\* find\_Repository(const string& repoName)

{

Repository\* temp = repositories;

while (temp != nullptr)

{

if (temp->name == repoName)

return temp;

temp = temp->next;

}

return nullptr;

}

int count\_Repositories()

{

int count = 0;

Repository\* temp = repositories;

while (temp != nullptr)

{

count++;

temp = temp->next;

}

return count;

}

int countFiles()

{

int count = 0;

Repository\* tempRepo = repositories;

while (tempRepo != nullptr)

{

File\* tempFile = tempRepo->files;

while (tempFile != nullptr)

{

count++;

tempFile = tempFile->next;

}

tempRepo = tempRepo->next;

}

return count;

}

int count\_Comments()

{

int count = 0;

Repository\* tempRepo = repositories;

while (tempRepo != nullptr)

{

Comment\* tempComment = tempRepo->comments;

while (tempComment != nullptr)

{

count++;

tempComment = tempComment->next;

}

tempRepo = tempRepo->next;

}

return count;

}

void delete\_Repository(const string& repoName)

{

Repository\* prevRepo = nullptr;

Repository\* currRepo = repositories;

// Search for the repository

while (currRepo != nullptr && currRepo->name != repoName)

{

prevRepo = currRepo;

currRepo = currRepo->next;

}

// If repository found, delete it

if (currRepo != nullptr)

{

if (prevRepo != nullptr)

{

prevRepo->next = currRepo->next;

}

else {

repositories = currRepo->next;

}

delete currRepo;

cout << "Repository '" << repoName << "' deleted successfully!\n";

}

else {

cout << "Repository not found.\n";

}

}

void saveDataToFile(const string& filename)

{

ofstream outfile(filename);

if (!outfile.is\_open())

{

cout << "Error: Unable to open file for writing.\n";

return;

}

outfile << username << "," << password << "," << followers << "," << count\_Repositories() << "\n";

Repository\* tempRepo = repositories;

while (tempRepo != nullptr)

{

outfile << tempRepo->name << "," << tempRepo->visibility << "\n";

File\* tempFile = tempRepo->files;

while (tempFile != nullptr)

{

outfile << "FILE," << tempFile->name << "," << tempFile->data << "\n";

tempFile = tempFile->next;

}

Comment\* tempComment = tempRepo->comments;

while (tempComment != nullptr)

{

outfile << "COMMENT," << tempComment->comment << "\n";

tempComment = tempComment->next;

}

outfile << "EOF\n"; // Mark the end of repository data

tempRepo = tempRepo->next;

}

outfile.close();

}

void loadDataFromFile(const string& filename)

{

ifstream infile(filename);

if (!infile.is\_open())

{

cout << "Error: Unable to open file for reading.\n";

return;

}

string line;

while (getline(infile, line))

{

istringstream iss(line);

string token;

getline(iss, token, ',');

if (token.empty())

continue;

username = token;

getline(iss, password, ',');

getline(iss, token, ',');

followers = stoi(token);

getline(iss, token, ',');

int numRepos = stoi(token);

for (int i = 0; i < numRepos; ++i)

{

string repoName, visibilityStr;

bool visibility;

getline(infile, line);

istringstream issRepo(line);

getline(issRepo, repoName, ',');

getline(issRepo, visibilityStr, ',');

visibility = (visibilityStr == "1" ? true : false);

add\_Repository(repoName, visibility);

Repository\* repo = find\_Repository(repoName);

if (repo)

{

while (true)

{

getline(infile, line);

istringstream issData(line);

getline(issData, token, ',');

if (token == "EOF") // Check for end of repository data

break;

if (token == "FILE") {

string fileName, fileData;

getline(issData, fileName, ',');

getline(issData, fileData, ',');

repo->add\_File(fileName, fileData);

}

else if (token == "COMMENT") {

string commentText;

getline(issData, commentText, ',');

repo->add\_Comment(commentText);

}

}

}

}

}

infile.close();

}

void display\_AccountDetails()

{

cout << "\t\t\t'" << username << "' Account Details " << endl;

cout << "\t\tFollowers: " << followers << endl;

cout << "\t\tRepositories: " << count\_Repositories() << endl;

cout << "\t\tFiles: " << countFiles() << endl;

cout << "\t\tComments: " << count\_Comments() << endl;

}

};

// Function declarations for additional functionality

void follow\_User(User& currentUser, User& otherUser);

void unfollow\_User(User& currentUser, User& otherUser);

void display\_Following(User& currentUser);

void register\_User(const string& username, const string& password)

{

ofstream outfile("users.csv", ios::app);

outfile << username << "," << password << ",0,0\n";

outfile.close();

cout << "User registered successfully!\n";

}

bool login\_User(const string& username, const string& password, User& user)

{

ifstream infile("users.csv");

string storedUsername, storedPassword;

int followers, repositories;

while (getline(infile, storedUsername, ','))

{

getline(infile, storedPassword, ',');

infile >> followers;

infile.ignore();

infile >> repositories;

infile.ignore();

if (storedUsername == username && storedPassword == password)

{

cout << "Login successful!\n";

user.username = username;

user.password = password;

user.followers = followers;

user.loadDataFromFile(username + ".csv");

infile.close();

return true;

}

// Skip the rest of the line

// infile.ignore(numeric\_limits<streamsize>::max(), '\n');

}

cout << "Invalid username or password. Please try again.\n";

infile.close();

return false;

}

void add\_Repository(User& user)

{

string repoName;

bool visib;

cout << "Enter repository name: ";

cin >> repoName;

cout << "Enter repository Visibility('0' for private & '1' for public): ";

cin >> visib;

user.add\_Repository(repoName, visib);

cout << "Repository added successfully!\n";

}

void add\_FileToRepository(User& user)

{

string repoName, fileName, fileData;

cout << "Enter repository name: ";

cin >> repoName;

Repository\* repo = user.find\_Repository(repoName);

if (repo != nullptr)

{

cout << "Enter file name: ";

cin >> fileName;

cout << "Enter file data: ";

cin.ignore();

getline(cin, fileData);

repo->add\_File(fileName, fileData);

cout << "File added to repository successfully!\n";

}

else

{

cout << "Repository not found.\n";

}

}

void add\_CommentToFile(User& user)

{

string repoName, fileName, commentText;

cout << "Enter repository name: ";

cin >> repoName;

Repository\* repo = user.find\_Repository(repoName);

if (repo != nullptr)

{

cout << "Enter file name: ";

cin >> fileName;

cout << "Enter comment: ";

cin.ignore();

getline(cin, commentText);

repo->add\_Comment(commentText);

cout << "Comment added to file successfully!\n";

}

else

{

cout << "Repository not found.\n";

}

}

void show\_RepositoryDetails(User& user)

{

string repoName;

cout << "Enter repository name: ";

cin >> repoName;

Repository\* repo = user.find\_Repository(repoName);

if (repo != nullptr)

{

cout << "Repository Name: " << repo->name << endl;

cout << "Files:\n";

File\* file = repo->files;

while (file != nullptr)

{

cout << "File Name: " << file->name << endl;

cout << "Data: " << file->data << endl;

file = file->next;

}

cout << "Comments:\n";

Comment\* comment = repo->comments;

while (comment != nullptr)

{

cout << "Comment: " << comment->comment << endl;

comment = comment->next;

}

}

else

{

cout << "Repository not found.\n";

}

}

void show\_AllUserFiles(User& user)

{

cout << "Files for User: " << user.username << endl;

Repository\* tempRepo = user.repositories;

while (tempRepo != nullptr)

{

cout << "Repository: " << tempRepo->name << endl;

File\* tempFile = tempRepo->files;

while (tempFile != nullptr)

{

cout << "File Name: " << tempFile->name << endl;

tempFile = tempFile->next;

}

tempRepo = tempRepo->next;

}

}

void display\_AccountDetails(User& user)

{

user.display\_AccountDetails();

}

void fork\_Repository(User& user, User& owner, const string& repoName)

{

Repository\* repoToCopy = owner.find\_Repository(repoName);

if (repoToCopy == nullptr)

{

cout << "Repository found.\n";

return;

}

// Create a new repository for the current user

user.add\_Repository(repoName, repoToCopy->visibility);

Repository\* newRepo = user.find\_Repository(repoName);

// Copy files from the original repository

File\* fileToCopy = repoToCopy->files;

while (fileToCopy != nullptr)

{

newRepo->add\_File(fileToCopy->name, fileToCopy->data);

fileToCopy = fileToCopy->next;

}

// Copy comments from the original repository

Comment\* commentToCopy = repoToCopy->comments;

while (commentToCopy != nullptr)

{

newRepo->add\_Comment(commentToCopy->comment);

commentToCopy = commentToCopy->next;

}

cout << "Repository '" << repoName << "' forked successfully!\n";

}

void follow\_User(User& currentUser, User& otherUser)

{

currentUser.follow(otherUser);

}

void unfollow\_User(User& currentUser, User& otherUser)

{

currentUser.unfollow(otherUser);

}

void display\_Following(User& currentUser)

{

currentUser.display\_Following();

}

void welcome()

{

cout << "\n\n";

cout << "\t\t\t\t---------------------------------------" << endl;

cout << "\t\t\t\t| [Data Structure Lab Final Project] | " << endl << endl;

cout << "\t\t\t\t| [1. Abdullah Zain and 2. Adnan Ali] | " << endl;

cout << "\t\t\t\t---------------------------------------" << endl << endl << endl;

cout << "\t======================== GITHUB ============================" << endl;

cout << "\n\n";

cout << "\n\n";

cout << "\n\n";

cout << "\n\n";

cout << "\t\t\t\t \* \* \* \*\*\*\*\*\*\*\*\* \* \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* \* \* \*\*\*\*\*\*\*\* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \* \* \*\*\*\*\*\*\*\*\* \* \* \* \* \* \* \* \* \*\*\*\*\*\*\*\* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \* \* \* \* \* \* \* \* \* \* \n";

cout << "\t\t\t\t \* \* \*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* \* \* \*\*\*\*\*\*\*\*\* \n";

cout << "\n\n\n\n\t\t\t\t Loading";

for (int i = 0; i <= 8; i++)

{

cout << ".\*.";

Sleep(1500);

}

system("cls");

}

void welcome1()

{

cout << "\t\t\t\t---------------------------------------" << endl;

cout << "\t\t\t\t| [Data Structure Lab Final Project] | " << endl << endl;

cout << "\t\t\t\t| [1. Abdullah Zain and 2. Adnan Ali] | " << endl;

cout << "\t\t\t\t---------------------------------------" << endl << endl << endl;

cout << "\t======================== GITHUB============================" << endl;

Sleep(10);

//system("cls");

}

int main()

{

int choice;

string username, password;

welcome();

welcome1();

User owner("owner\_username", "owner\_password");

while (true)

{

cout << "Welcome to the User Management System\n";

cout << "1. Register\n";

cout << "2. Login\n";

cout << "3. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

if (choice == 1)

{

cout << "Enter username: ";

cin >> username;

cout << "Enter password: ";

cin >> password;

register\_User(username, password);

}

else if (choice == 2)

{

cout << "Enter username: ";

cin >> username;

cout << "Enter password: ";

cin >> password;

if (login\_User(username, password, owner))

{

int userChoice;

owner.display\_AccountDetails();

while (true)

{

cout << "1. Add Repository\n";

cout << "2. Add File to Repository\n";

cout << "3. Add Comment to File\n";

cout << "4. Show Repository Details\n";

cout << "5. Show All User Files\n";

cout << "6. Delete Repository\n";

cout << "7. Follow User\n"; // New option

cout << "8. Unfollow User\n"; // New option

cout << "9. Display Following\n"; // New option

cout << "10. Logout\n";

cout << "Enter your choice: ";

cin >> userChoice;

if (userChoice == 1)

{

add\_Repository(owner);

}

else if (userChoice == 2)

{

add\_FileToRepository(owner);

}

else if (userChoice == 3)

{

add\_CommentToFile(owner);

}

else if (userChoice == 4)

{

show\_RepositoryDetails(owner);

}

else if (userChoice == 5)

{

show\_AllUserFiles(owner);

}

else if (userChoice == 6)

{

string repoName;

cout << "Enter the name of the repository you want to delete: ";

cin >> repoName;

owner.delete\_Repository(repoName);

}

else if (userChoice == 7)

{

follow\_User(owner, owner); // Follow yourself

}

else if (userChoice == 8)

{

unfollow\_User(owner, owner); // Unfollow yourself

}

else if (userChoice == 9)

{

display\_Following(owner); // Display users being followed

}

else if (userChoice == 10)

{

cout << "Logging out...\n";

owner.saveDataToFile(username + ".csv");

break;

}

else

{

cout << "Invalid choice. Please try again.\n";

}

}

break;

}

}

else if (choice == 3)

{

cout << "Exiting program...\n";

return 0;

}

else {

cout << "Invalid choice. Please try again.\n";

}

}

return 0;

}

**Screenshot:**





