**GitHub Simulation Console Application Report**

**Project: GitHub Simulation**

**Iqra Iqbal**

**22f-3087**

**BSAI-4A**

**HEADER**

// PROJECT.h

#ifndef PROJECT\_H

#define PROJECT\_H

#include <iostream>

#include <string>

#include <iomanip>

#include <thread>

#include <chrono>

using namespace std;

class HashTable {

private:

static const int TABLE\_SIZE = 1000;

struct Repository {

string name;

bool isPublic;

string\* files;

int fileCount;

string\* commits;

int commitCount;

Repository(const string& repoName);

~Repository();

void addFile(const string& fileName);

void removeFile(const string& fileName);

void addCommit(const string& commitMessage);

};

struct UserRecord {

string username;

string password;

Repository\*\* repositories;

int repositoryCount;

string\* followers[TABLE\_SIZE];

int followerCount;

string\* following[TABLE\_SIZE];

int followingCount;

UserRecord(const string& uname, const string& pwd);

bool follows(const string& otherUsername);

};

UserRecord\* table[TABLE\_SIZE];

int hashFunction(const string& key);

public:

HashTable();

void addUser(const string& username, const string& password);

bool login(const string& username, const string& password);

void addRepository(const string& username, const string& repositoryName);

void deleteRepository(const string& username, const string& repositoryName);

void removeFile(const string& username, const string& repositoryName, const string& fileName);

void setRepositoryVisibility(const string& username, const string& repositoryName, bool isPublic);

void addFile(const string& username, const string& repositoryName, const string& fileName);

void listRepositories(const string& username);

void commit(const string& username, const string& repositoryName, const string& commitMessage);

void forkRepository(const string& sourceUsername, const string& sourceRepoName, const string& destinationUsername);

void followUser(const string& username, const string& otherUsername);

void unfollowUser(const string& username, const string& otherUsername);

void viewProfile(const string& username);

~HashTable();

};

#endif // PROJECT\_H

#pragma once

**CPP**

// PROJECT.cpp

#include "PROJECT.h"

////.................constructor.................\\\\

HashTable::Repository::Repository(const string& repoName) {

name = repoName;

isPublic = false;

fileCount = 0;

commitCount = 0;

files = new string[TABLE\_SIZE];

commits = new string[TABLE\_SIZE];

}

////.................Destructor.................\\\\

HashTable::Repository::~Repository() {

delete[] files;

delete[] commits;

}

////.................Function for AddFile.................\\\\

void HashTable::Repository::addFile(const string& fileName) {

files[fileCount++] = fileName;

}

////.................Function for RemoveFile.................\\\\

void HashTable::Repository::removeFile(const string& fileName) {

for (int i = 0; i < fileCount; i++) {

if (files[i] == fileName) {

for (int j = i; j < fileCount - 1; ++j) {

files[j] = files[j + 1];

}

fileCount--;

break;

}

}

}

////.................Function for Addcommit.................\\\\

void HashTable::Repository::addCommit(const string& commitMessage) {

commits[commitCount++] = commitMessage;

}

////.................Function for userRecord.................\\\\

HashTable::UserRecord::UserRecord(const string& uname, const string& pwd) {

username = uname;

password = pwd;

repositoryCount = 0;

followerCount = 0;

followingCount = 0;

repositories = new Repository \* [TABLE\_SIZE];

for (int i = 0; i < TABLE\_SIZE; i++) {

repositories[i] = nullptr;

followers[i] = nullptr;

following[i] = nullptr;

}

}

////.................Function for follow.................\\\\

bool HashTable::UserRecord::follows(const string& otherUsername) {

for (int i = 0; i < followingCount; i++) {

if (\*following[i] == otherUsername) {

return true;

}

}

return false;

}

////.................constructor.................\\\\

HashTable::HashTable() {

for (int i = 0; i < TABLE\_SIZE; i++) {

table[i] = nullptr;

}

}

////.................HASH-FUNCTION.................\\\\

int HashTable::hashFunction(const string& key) {

int hash = 0;

for (char c : key) {

hash += c;

}

return hash % TABLE\_SIZE;

}

////.................Function for ADD-USER.................\\\\

void HashTable::addUser(const string& username, const string& password) {

int index = hashFunction(username);

table[index] = new UserRecord(username, password);

}

////.................Function for login.................\\\\

bool HashTable::login(const string& username, const string& password) {

int index = hashFunction(username);

if (table[index] != nullptr && table[index]->username == username && table[index]->password == password) {

return true;

}

return false;

}

////.................Function for Add-Repository.................\\\\

void HashTable::addRepository(const string& username, const string& repositoryName) {

int index = hashFunction(username);

if (table[index] != nullptr && table[index]->username == username) {

table[index]->repositories[table[index]->repositoryCount++] = new Repository(repositoryName);

}

}

////.................Function for Delete repository.................\\\\

void HashTable::deleteRepository(const string& username, const string& repositoryName) {

int index = hashFunction(username);

if (table[index] != nullptr && table[index]->username == username) {

for (int i = 0; i < table[index]->repositoryCount; i++) {

if (table[index]->repositories[i]->name == repositoryName) {

delete table[index]->repositories[i];

for (int j = i; j < table[index]->repositoryCount - 1; ++j) {

table[index]->repositories[j] = table[index]->repositories[j + 1];

}

table[index]->repositoryCount--;

break;

}

}

}

}

////.................Function for remove File.................\\\\

void HashTable::removeFile(const string& username, const string& repositoryName, const string& fileName) {

int userIndex = hashFunction(username);

if (table[userIndex] != nullptr && table[userIndex]->username == username) {

for (int i = 0; i < table[userIndex]->repositoryCount; i++) {

if (table[userIndex]->repositories[i]->name == repositoryName) {

table[userIndex]->repositories[i]->removeFile(fileName);

break;

}

}

}

}

////.................Function for set visibility.................\\\\

void HashTable::setRepositoryVisibility(const string& username, const string& repositoryName, bool isPublic) {

int index = hashFunction(username);

if (table[index] != nullptr && table[index]->username == username) {

for (int i = 0; i < table[index]->repositoryCount; i++) {

if (table[index]->repositories[i]->name == repositoryName) {

table[index]->repositories[i]->isPublic = isPublic;

break;

}

}

}

}

////.................Function for AddFile.................\\\\

void HashTable::addFile(const string& username, const string& repositoryName, const string& fileName) {

int userIndex = hashFunction(username);

if (table[userIndex] != nullptr && table[userIndex]->username == username) {

for (int i = 0; i < table[userIndex]->repositoryCount; i++) {

if (table[userIndex]->repositories[i]->name == repositoryName) {

table[userIndex]->repositories[i]->addFile(fileName);

break;

}

}

}

}

////.................Function for list Repository .................\\\\

void HashTable::listRepositories(const string& username) {

int index = hashFunction(username);

if (table[index] != nullptr && table[index]->username == username) {

for (int i = 0; i < table[index]->repositoryCount; i++) {

cout << table[index]->repositories[i]->name << endl;

}

}

}

////.................Function for commit .................\\\\

void HashTable::commit(const string& username, const string& repositoryName, const string& commitMessage) {

int userIndex = hashFunction(username);

if (table[userIndex] != nullptr && table[userIndex]->username == username) {

for (int i = 0; i < table[userIndex]->repositoryCount; i++) {

if (table[userIndex]->repositories[i]->name == repositoryName) {

table[userIndex]->repositories[i]->addCommit(commitMessage);

break;

}

}

}

}

////.................Function for fork repository .................\\\\

void HashTable::forkRepository(const string& sourceUsername, const string& sourceRepoName, const string& destinationUsername) {

int sourceIndex = hashFunction(sourceUsername);

int destIndex = hashFunction(destinationUsername);

if (table[sourceIndex] != nullptr && table[sourceIndex]->username == sourceUsername) {

for (int i = 0; i < table[sourceIndex]->repositoryCount; i++) {

if (table[sourceIndex]->repositories[i]->name == sourceRepoName) {

table[destIndex]->repositories[table[destIndex]->repositoryCount++] = table[sourceIndex]->repositories[i];

break;

}

}

}

}

////.................Function for Follow user .................\\\\

void HashTable::followUser(const string& username, const string& otherUsername) {

int userIndex = hashFunction(username);

int otherIndex = hashFunction(otherUsername);

if (table[userIndex] != nullptr && table[userIndex]->username == username && table[otherIndex] != nullptr && table[otherIndex]->username == otherUsername) {

table[userIndex]->following[table[userIndex]->followingCount++] = &table[otherIndex]->username;

table[otherIndex]->followers[table[otherIndex]->followerCount++] = &table[userIndex]->username;

}

}

////.................Function for unfollow user.................\\\\

void HashTable::unfollowUser(const string& username, const string& otherUsername) {

int userIndex = hashFunction(username);

if (table[userIndex] != nullptr && table[userIndex]->username == username) {

for (int i = 0; i < table[userIndex]->followingCount; i++) {

if (\*table[userIndex]->following[i] == otherUsername) {

for (int j = i; j < table[userIndex]->followingCount - 1; ++j) {

table[userIndex]->following[j] = table[userIndex]->following[j + 1];

}

table[userIndex]->followingCount--;

break;

}

}

}

}

////.................Function for View user profile.................\\\\

void HashTable::viewProfile(const string& username) {

int index = hashFunction(username);

if (table[index] != nullptr && table[index]->username == username) {

cout << "Username: " << table[index]->username << endl;

cout << "Repositories: " << table[index]->repositoryCount << endl;

cout << "Followers: " << table[index]->followerCount << endl;

cout << "Following: " << table[index]->followingCount << endl;

}

}

////.................Destructor .................\\\\

HashTable::~HashTable() {

for (int i = 0; i < TABLE\_SIZE; i++) {

if (table[i] != nullptr) {

delete table[i];

}

}

}

**MAIN**

#include <iostream>

#include <string>

#include <iomanip>

#include <thread>

#include <chrono>

#include "PROJECT.h"

using namespace std;

int main() {

HashTable users;

int choice;

string username, password, repositoryName, fileName, commitMessage, sourceUsername, sourceRepoName, destinationUsername;

cout << endl << endl << endl << endl << endl << endl << endl << endl << endl ;

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

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

cout << "\t\t\t\t\t\* \*" << endl;

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

cout<< "\t\t\t\t\t\*\t\t22F-3087 \*\n";

cout << "\t\t\t\t\t\*\t\tProject: GitHub Simulation \*\n";

cout << "\t\t\t\t\t\* \*" << endl;

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

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;;

this\_thread::sleep\_for(chrono::seconds(5));

system("cls");

do {

cout << "\n========================================================================================================================" << endl << endl;

cout << "\t\t\t\t\t\t\tMenu " << endl << endl;

cout << "\n========================================================================================================================" << endl << endl;

cout << "\t\t\t\t\t\t1.\tAdd User" << endl;

cout << "\t\t\t\t\t\t2.\tLogin" << endl;

cout << "\t\t\t\t\t\t3.\tAdd Repository" << endl;

cout << "\t\t\t\t\t\t4.\tDelete Repository" << endl;

cout << "\t\t\t\t\t\t5.\tRemove File" << endl;

cout << "\t\t\t\t\t\t6.\tSet Repository Visibility" << endl;

cout << "\t\t\t\t\t\t7.\tAdd File" << endl;

cout << "\t\t\t\t\t\t8.\tList Repositories" << endl;

cout << "\t\t\t\t\t\t9.\tCommit" << endl;

cout << "\t\t\t\t\t\t10.\tFork Repository" << endl;

cout << "\t\t\t\t\t\t11.\tFollow User" << endl;

cout << "\t\t\t\t\t\t12.\tUnfollow User" << endl;

cout << "\t\t\t\t\t\t13.\tView Profile" << endl;

cout << "\t\t\t\t\t\t0.\tExit" << endl << endl;

cout << "\n========================================================================================================================" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Add User" << endl;

cout << "Enter username: ";

cin >> username;

cout << "Enter password: ";

cin >> password;

users.addUser(username, password);

cout << "User added successfully!" << endl;

cout << "\n========================================================================================================================" << endl;

break;

case 2:

cout << "Login" << endl;

cout << "Enter username: ";

cin >> username;

cout << "Enter password: ";

cin >> password;

if (users.login(username, password)) {

cout << "Login successful!" << endl;

}

else {

cout << "Login failed. Invalid username or password." << endl;

}

cout << "\n========================================================================================================================" << endl;

break;

case 3:

cout << "Add Repository" << endl;

cout << "Enter username: ";

cin >> username;

cout << "Enter repository name: ";

cin >> repositoryName;

users.addRepository(username, repositoryName);

cout << "Repository added successfully!" << endl;

cout << "\n========================================================================================================================" << endl;

break;

case 4:

cout << "Delete Repository" << endl;

cout << "Enter username: ";

cin >> username;

cout << "Enter repository name: ";

cin >> repositoryName;

users.deleteRepository(username, repositoryName);

cout << "Repository deleted successfully!" << endl;

cout << "\n========================================================================================================================" << endl;

break;

case 5:

cout << "Remove File" << endl;

cout << "Enter username: ";

cin >> username;

cout << "Enter repository name: ";

cin >> repositoryName;

cout << "Enter file name: ";

cin >> fileName;

users.removeFile(username, repositoryName, fileName);

cout << "File removed successfully!" << endl;

cout << "\n========================================================================================================================" << endl;

break;

case 6:

cout << "Set Repository Visibility" << endl;

cout << "Enter username: ";

cin >> username;

cout << "Enter repository name: ";

cin >> repositoryName;

cout << "Set repository visibility (1 for public, 0 for private): ";

bool isPublic;

cin >> isPublic;

users.setRepositoryVisibility(username, repositoryName, isPublic);

cout << "Repository visibility set successfully!" << endl;

cout << "\n========================================================================================================================" << endl;

break;

case 7:

cout << "Add File" << endl;

cout << "Enter username: ";

cin >> username;

cout << "Enter repository name: ";

cin >> repositoryName;

cout << "Enter file name: ";

cin >> fileName;

users.addFile(username, repositoryName, fileName);

cout << "File added successfully!" << endl;

cout << "\n========================================================================================================================" << endl;

break;

case 8:

cout << "List Repositories" << endl;

cout << "Enter username: ";

cin >> username;

users.listRepositories(username);

cout << "========================================================================================================================" << endl;

break;

case 9:

cout << "Commit" << endl;

cout << "Enter username: ";

cin >> username;

cout << "Enter repository name: ";

cin >> repositoryName;

cout << "Enter commit message: ";

cin.ignore();

getline(cin, commitMessage);

users.commit(username, repositoryName, commitMessage);

cout << "Commit successful!" << endl;

cout << "\n========================================================================================================================" << endl;

break;

case 10:

cout << "Fork Repository" << endl;

cout << "Enter source username: ";

cin >> sourceUsername;

cout << "Enter source repository name: ";

cin >> sourceRepoName;

cout << "Enter destination username: ";

cin >> destinationUsername;

users.forkRepository(sourceUsername, sourceRepoName, destinationUsername);

cout << "Repository forked successfully!" << endl;

cout << "\n========================================================================================================================" << endl;

break;

case 11:

cout << "Follow User" << endl;

cout << "Enter username: ";

cin >> username;

cout << "Enter username to follow: ";

cin >> destinationUsername;

users.followUser(username, destinationUsername);

cout << "User followed successfully!" << endl;

cout << "\n========================================================================================================================" << endl;

break;

case 12:

cout << "Unfollow User" << endl;

cout << "Enter username: ";

cin >> username;

cout << "Enter username to unfollow: ";

cin >> destinationUsername;

users.unfollowUser(username, destinationUsername);

cout << "User unfollowed successfully!" << endl;

cout << "\n========================================================================================================================" << endl;

break;

case 13:

cout << "View Profile" << endl;

cout << "Enter username: ";

cin >> username;

users.viewProfile(username);

cout << "\n========================================================================================================================" << endl;

break;

default:

cout << "Invalid Choice" << endl;

cout << "Please enter a number between 0 and 13." << endl;

cout << "\n========================================================================================================================" << endl;

break;

}

} while (choice != 0);

return 0;

}

**output**







 



