**Data Structures Lab**

**(CL-2005)**



**Project – GitHub Simulation**

5th May, 2024

|  |  |  |
| --- | --- | --- |
| Abdul Hadi | 22F-3582 | BCS-4D |
| Ehsan | 22F-8753 | BCS-4D |
| Zain | 22F-3247 | BCS-4D |

**NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES, CHINIOT-**

**FAISALABAD**

**TABLE OF CONTENT**

|  |  |
| --- | --- |
| **1** | Introduction |
| **2** | Objectives |
| **3** | Project Description |
| **4** | Key Features |
| **5** | GitHub Link |
| **6** | Code |
| **7** | Outputs |

**Introduction:**

The following document presents an exploration into a console application written in C++, simulating the functionality of GitHub. This application aims to provide users with a platform for managing repositories, commits, files, and user interactions, all within a command-line interface environment.

**Project Description:**

Experience GitHub's core features from your console with this simulation project. Create, manage, and collaborate on coding projects effortlessly. From creating repositories to adding commits and files, everything is simplified for efficient project management. Interact with other users by following and unfollowing them, and explore their profiles to discover their projects. Behind the scenes, sophisticated coding techniques ensure a smooth and enjoyable experience. Whether you're a seasoned coder or just starting out, this C++ GitHub simulation makes coding easier and more enjoyable.

**Objectives:**

1. **Functional Simulation:** The primary objective of this project is to create a functional simulation of GitHub, allowing users to perform common actions such as creating accounts, logging in, creating repositories, adding commits and files, and interacting with other users through follows and unfollows.
2. **User Management:** Efficient user management capabilities are essential for any platform. This project aims to implement robust user registration, login, and profile viewing functionalities.
3. **Repository Management:** Providing users with the ability to create repositories, add commits, and manage files within these repositories is a core feature. The application seeks to mimic GitHub's repository management functionalities.
4. **Social Interaction:** The project includes features for users to follow and unfollow other users, fostering a sense of community and collaboration within the simulated environment.

Top of Form

**Key Features:**

1. **User Authentication:** Users can register accounts with unique usernames and passwords, enabling secure access to the application.
2. **Repository Creation:** Users have the capability to create repositories, each with its own set of commits and files.
3. **Commit and File Management:** Adding commits with messages and files to repositories allows for version control and project organization.
4. **User Interaction:** Users can follow and unfollow other users, facilitating social interaction and collaboration.
5. **Profile Viewing:** Users can view their own and others' profiles, including follower lists and repository details.

**GitHub Link:**

<https://github.com/MalikEhsan00/projectt/blob/a375e5d1fe53db5c1cf99208327d5b6eaa645d65/proj%20(1).docx>

**Code:**

// HEADER FILE

#pragma once

#include <iostream>

#include <string>

#include <iomanip>

#include <fstream>

#include <sstream>

using namespace std;

const int MAX\_USERS = 100;

const int MAX\_FOLLOWERS = 100;

const int MAX\_REPOSITORIES = 100;

const int MAX\_COMMITS = 100;

const int MAX\_FILES = 100;

class Commit;

class File;

class Repository;

class Tree;

class User;

class UserRepository;

class SocialGraph;

class UserManager;

// SOURCE FILE

#include "github.h"

class Commit {

private:

string message;

public:

Commit(string msg) {

message = msg;

}

string getMessage() const {

return message;

}

};

class File {

private:

string name;

public:

File(string filename) {

name = filename;

}

string getName() const {

return name;

}

};

class Repository {

private:

string name;

Commit\* commits[MAX\_COMMITS];

File\* files[MAX\_FILES];

int commitCount;

int fileCount;

public:

Repository(string repoName) {

name = repoName;

commitCount = 0;

fileCount = 0;

}

string getName() const {

return name;

}

bool addCommit(Commit\* commit) {

if (commitCount < MAX\_COMMITS) {

commits[commitCount++] = commit;

return true;

}

return false;

}

bool addFile(File\* file) {

if (fileCount < MAX\_FILES) {

files[fileCount++] = file;

return true;

}

return false;

}

const Commit\*\* getCommits() const {

return (const Commit\*\*)commits;

}

const File\*\* getFiles() const {

return (const File\*\*)files;

}

int getCommitCount() const {

return commitCount;

}

int getFileCount() const {

return fileCount;

}

};

class Tree {

private:

struct TreeNode {

Repository\* repository;

TreeNode\* left;

TreeNode\* right;

TreeNode(Repository\* repo) {

repository = repo;

left = nullptr;

right = nullptr;

}

};

TreeNode\* root;

void deleteTree(TreeNode\* node) {

if (node) {

deleteTree(node->left);

deleteTree(node->right);

delete node->repository;

delete node;

}

}

void savefile(TreeNode\* node, fstream& file) {

if (node == nullptr) { return; }

savefile(node->left, file);

file << node->repository->getName() << ",";

int total\_file = node->repository->getFileCount();

const File\*\* files = node->repository->getFiles();

for (int i = 0; i < total\_file; i++)

{

file << files[i]->getName();

if (i + 1 != total\_file) {

file << ":";

}

}

file << ",";

int total\_commits = node->repository->getCommitCount();

const Commit\*\* commits = node->repository->getCommits();

for (int i = 0; i < total\_commits; i++)

{

file << commits[i]->getMessage();

if (i + 1 != total\_commits) {

file << ":";

}

}

file << endl;

savefile(node->right, file);

}

public:

Tree() {

root = nullptr;

}

~Tree() {

deleteTree(root);

}

bool addRepository(Repository\* repository) {

TreeNode\* newNode = new TreeNode(repository);

if (!root) {

root = newNode;

return true;

}

TreeNode\* current = root;

TreeNode\* parent = nullptr;

while (current) {

parent = current;

if (repository->getName() < current->repository->getName()) {

current = current->left;

}

else if (repository->getName() > current->repository->getName()) {

current = current->right;

}

else {

delete newNode;

return false;

}

}

if (repository->getName() < parent->repository->getName()) {

parent->left = newNode;

}

else {

parent->right = newNode;

}

return true;

}

Repository\* searchRepository(const string& repoName) const {

TreeNode\* current = root;

while (current) {

if (repoName == current->repository->getName()) {

return current->repository;

}

else if (repoName < current->repository->getName()) {

current = current->left;

}

else {

current = current->right;

}

}

return nullptr;

}

void SaveData() {

fstream file;

file.open("TreeSaveFile.csv", ios::out);

if (!file.is\_open()) { cout << "File: Error\n"; return; }

savefile(root, file);

file.close();

}

void LoadData() {

fstream file;

file.open("TreeSaveFile.csv", ios::in);

if (!file.is\_open()) { cout << "File: Error\n"; return; }

string line;

while (getline(file, line)) {

stringstream ss, ssfile, sscommit;

string repos, all\_files, file\_name, all\_commits, commit\_msg;

ss << line;

try

{

getline(ss, repos, ',');

Repository\* repo = new Repository(repos);

this->addRepository(repo);

getline(ss, all\_files, ',');

ssfile << all\_files;

while (getline(ssfile, file\_name, ':')) {

File\* add\_file = new File(file\_name);

repo->addFile(add\_file);

}

getline(ss, all\_commits);

sscommit << all\_commits;

while (getline(sscommit, commit\_msg, ':')) {

Commit\* add\_commit = new Commit(commit\_msg);

repo->addCommit(add\_commit);

}

}

catch (const std::exception& ex)

{

cout << ex.what();

}

}

file.close();

}

};

class User {

private:

string username;

string password;

string followers[MAX\_FOLLOWERS];

int followerCount;

public:

User() {

username = "";

password = "";

followerCount = 0;

}

User(string uname, string pwd) {

username = uname;

password = pwd;

followerCount = 0;

}

string getUsername() const {

return username;

}

string getPassword() const {

return password;

}

const string\* getFollowers() const {

return followers;

}

int getFollowerCount() const {

return followerCount;

}

bool addFollower(const string& follower) {

if (followerCount < MAX\_FOLLOWERS) {

followers[followerCount++] = follower;

return true;

}

return false;

}

bool removeFollower(const string& follower) {

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

if (followers[i] == follower) {

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

followers[j] = followers[j + 1];

}

followerCount--;

return true;

}

}

return false;

}

};

class UserRepository {

private:

string name;

string commits[MAX\_FOLLOWERS];

string files[MAX\_FOLLOWERS];

int commitCount;

int fileCount;

public:

UserRepository(string repoName) {

repoName = name;

commitCount = 0;

fileCount = 0;

}

string getName() const {

return name;

}

const string\* getCommits() const {

return commits;

}

const string\* getFiles() const {

return files;

}

int getCommitCount() const {

return commitCount;

}

int getFileCount() const {

return fileCount;

}

bool addCommit(const string& commit) {

if (commitCount < MAX\_FOLLOWERS) {

commits[commitCount++] = commit;

return true;

}

return false;

}

bool addFile(const string& file) {

if (fileCount < MAX\_FOLLOWERS) {

files[fileCount++] = file;

return true;

}

return false;

}

};

class SocialGraph {

private:

string users[MAX\_USERS];

int userCount;

int followMatrix[MAX\_USERS][MAX\_USERS];

public:

SocialGraph() : userCount(0) {

for (int i = 0; i < MAX\_USERS; ++i) {

for (int j = 0; j < MAX\_USERS; ++j) {

followMatrix[i][j] = 0;

}

}

}

bool addUser(const string& username) {

if (userCount < MAX\_USERS) {

users[userCount++] = username;

return true;

}

return false;

}

bool followUser(int followerIndex, int followeeIndex) {

if (followerIndex < userCount && followeeIndex < userCount) {

followMatrix[followerIndex][followeeIndex] = 1;

return true;

}

return false;

}

bool unfollowUser(int followerIndex, int followeeIndex) {

if (followerIndex < userCount && followeeIndex < userCount) {

followMatrix[followerIndex][followeeIndex] = 0;

return true;

}

return false;

}

const string\* getUsernames() const {

return users;

}

int getUserCount() const {

return userCount;

}

const int\* getFollowMatrix() const {

return \*followMatrix;

}

void SaveData() {

fstream file;

file.open("MatrixSaveFile.csv", ios::out);

if (!file.is\_open()) { cout << "File: Error\n"; return; }

for (int i = 0; i < MAX\_USERS; i++)

{

for (int j = 0; j < MAX\_USERS; j++)

{

file << followMatrix[i][j];

if (j + 1 != MAX\_USERS) {

file << ",";

}

}

file << endl;

}

file.close();

}

void LoadData(User\* users, int user\_count) {

for (int i = 0; i < user\_count; i++) {

this->addUser(users[i].getUsername());

}

fstream file;

file.open("MatrixSaveFile.csv", ios::in);

if (!file.is\_open()) { cout << "File: Error\n"; return; }

stringstream ss, matrix\_line;

string line, status;

int i = 0, j;

while (getline(file, line)) {

ss << line;

i++;

j = 0;

while (getline(ss, status, ',')) {

j++;

if (status == "1") {

followMatrix[i][j] = 1;

}

}

}

file.close();

}

};

class UserManager {

private:

User users[MAX\_USERS];

int userCount;

public:

UserManager() {

userCount = 0;

for (int i = 0; i < MAX\_USERS; ++i) {

users[i] = User();

}

}

User\* getUsers() {

return users;

}

int getUserCount() {

return userCount;

}

bool registerUser(const string& username, const string& password) {

if (userCount < MAX\_USERS) {

users[userCount++] = User(username, password);

return true;

}

return false;

}

User\* loginUser(const string& username, const string& password) {

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

if (users[i].getUsername() == username && users[i].getPassword() == password) {

return &users[i];

}

}

return nullptr;

}

User\* getUser(const string& username) {

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

if (users[i].getUsername() == username) {

return &users[i];

}

}

return nullptr;

}

void SaveData() {

fstream file;

file.open("UserAccountData.csv", ios::out);

if (!file.is\_open()) { cout << "File: Error\n"; return; }

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

file << users[i].getUsername() << "," << users[i].getPassword() << ",";

int followers = users[i].getFollowerCount();

const string\* followers\_list = users[i].getFollowers();

for (int j = 0; j < followers; j++) {

file << followers\_list[j];

if (j + 1 != followers) {

file << ":";

}

}

file << endl;

}

file.close();

}

void LoadData() {

fstream file;

file.open("UserAccountData.csv", ios::in);

if (!file.is\_open()) { cout << "File: Error\n"; return; }

string line;

while (getline(file, line)) {

stringstream ss;

ss << line;

string username, password;

getline(ss, username, ',');

getline(ss, password, ',');

this->registerUser(username, password);

string followers\_list, follower\_name;

stringstream ssfollower;

getline(ss, followers\_list);

ssfollower << followers\_list;

while (getline(ssfollower, follower\_name, ':')) {

this->getUser(username)->addFollower(follower\_name);

}

}

file.close();

}

};

// MAIN FILE

#include "github.h"

#include "source.cpp"

int main() {

UserManager userManager;

SocialGraph socialGraph;

Tree repositoryTree;

repositoryTree.LoadData();

userManager.LoadData();

socialGraph.LoadData(userManager.getUsers(), userManager.getUserCount());

while (true) {

system("CLS");

cout << "\t\t\t-------------------------------------------\n";

cout << setw(20) << left << "" << "\t\t\tGITHUB MENU" << endl;

cout << "\t\t\t-------------------------------------------\n";

cout << "\t\t\t1. Create Account" << endl;

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

cout << "\t\t\t3. View Profile" << endl;

cout << "\t\t\t4. Follow User" << endl;

cout << "\t\t\t5. Unfollow User" << endl;

cout << "\t\t\t6. Create Repository" << endl;

cout << "\t\t\t7. Add Commit" << endl;

cout << "\t\t\t8. Add File" << endl;

cout << "\t\t\t9. View Repository Details" << endl;

cout << "\t\t\t10. Exit" << endl;

cout << "\t\t\t-------------------------------------------\n";

cout << "\t\t\tChoose an option: ";

int choice;

cin >> choice;

if (choice == 1) {

system("CLS");

cout << "\t\t\t-------------------------------------------\n";

cout << setw(20) << left << "" << "\t\t\tACCOUNT SIGNUP" << endl;

cout << "\t\t\t-------------------------------------------\n";

string username, password;

cout << "\n\t\t\tEnter username: ";

cin >> username;

cout << "\t\t\tEnter password: ";

cin >> password;

if (userManager.registerUser(username, password)) {

cout << "\n\t\t\tACCOUNT CREATED!" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

socialGraph.addUser(username);

}

else {

cout << "UNABLE TO CREATE AN ACCOUNT!" << endl;

cout << "\t\t\t-------------------------------------------\n";

}

}

else if (choice == 2) {

system("CLS");

cout << "\t\t\t-------------------------------------------\n";

cout << setw(20) << left << "" << "\t\t\tACCOUNT LOGIN" << endl;

cout << "\t\t\t-------------------------------------------\n";

string username, password;

cout << "\n\t\t\tEnter username: ";

cin >> username;

cout << "\t\t\tEnter password: ";

cin >> password;

User\* loggedInUser = userManager.loginUser(username, password);

if (loggedInUser) {

cout << "\n\t\t\tLOGIN SUCCESSFUL!" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

else {

cout << "\n\t\t\tINVALID USERNAME OR PASSWORD! TRY AGAIN" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

}

else if (choice == 3) {

system("CLS");

cout << "\t\t\t-------------------------------------------\n";

cout << setw(20) << left << "" << "\t\t\tYOUR PROFILE" << endl;

cout << "\t\t\t-------------------------------------------\n";

string username;

cout << "\n\t\t\tEnter username: ";

cin >> username;

User\* userProfile = userManager.getUser(username);

if (userProfile) {

cout << "\n\t\t\tUsername: " << userProfile->getUsername() << endl;

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

const string\* followers = userProfile->getFollowers();

for (int i = 0; i < userProfile->getFollowerCount(); ++i) {

cout << "\t\t\t- " << followers[i] << endl << endl;

}

cout << "\t\t\t-------------------------------------------\n";

}

else {

cout << "\n\t\t\tUSER NOT FOUND" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

}

else if (choice == 4) {

system("CLS");

cout << "\t\t\t-------------------------------------------\n";

cout << setw(20) << left << "" << "\t\t\tFOLLOW ACCOUNTS" << endl;

cout << "\t\t\t-------------------------------------------\n";

string followerName, followeeName;

cout << "\n\t\t\tEnter your username: ";

cin >> followerName;

cout << "\t\t\tFollow: ";

cin >> followeeName;

int followerIndex = -1, followeeIndex = -1;

for (int i = 0; i < socialGraph.getUserCount(); ++i) {

if (socialGraph.getUsernames()[i] == followerName) {

followerIndex = i;

}

if (socialGraph.getUsernames()[i] == followeeName) {

followeeIndex = i;

}

}

if (followerIndex != -1 && followeeIndex != -1) {

if (socialGraph.followUser(followerIndex, followeeIndex)) {

cout << "\n\t\t\tYou are now following " << followeeName << "." << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

userManager.getUser(followeeName)->addFollower(followerName);

}

else {

cout << "\n\t\t\tFAILED TO FOLLOW USER! TRY AGAIN" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

}

else {

cout << "\n\t\t\tINVALID USERNAME! TRY AGAIN" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

}

else if (choice == 5) {

system("CLS");

cout << "\t\t\t-------------------------------------------\n";

cout << setw(20) << left << "" << "\t\t\tUNFOLLOW ACCOUNTS" << endl;

cout << "\t\t\t-------------------------------------------\n";

string followerName, followeeName;

cout << "\n\t\t\tEnter your username: ";

cin >> followerName;

cout << "\t\t\tUnfollow: ";

cin >> followeeName;

int followerIndex = -1, followeeIndex = -1;

for (int i = 0; i < socialGraph.getUserCount(); ++i) {

if (socialGraph.getUsernames()[i] == followerName) {

followerIndex = i;

}

if (socialGraph.getUsernames()[i] == followeeName) {

followeeIndex = i;

}

}

if (followerIndex != -1 && followeeIndex != -1) {

if (socialGraph.unfollowUser(followerIndex, followeeIndex)) {

cout << "\n\t\t\tYou have unfollowed " << followeeName << "." << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

userManager.getUser(followeeName)->removeFollower(followerName);

}

else {

cout << "\n\t\t\tFAILED TO UNFOLLOW USER! TRY AGAIN" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

}

else {

cout << "\n\t\t\tINVALID USERNAME! TRY AGAIN" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

}

else if (choice == 6) {

system("CLS");

cout << "\t\t\t-------------------------------------------\n";

cout << setw(20) << left << "" << "\t\t\tCREATE REPOSITORY" << endl;

cout << "\t\t\t-------------------------------------------\n";

string repoName;

cout << "\n\t\t\tEnter repository name: ";

cin >> repoName;

Repository\* newRepo = new Repository(repoName);

if (repositoryTree.addRepository(newRepo)) {

cout << "\n\t\t\tREPOSITORY CREATED!" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

else {

cout << "\n\t\t\tSAME NAME ALREADY EXISTS!" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

delete newRepo;

}

}

else if (choice == 7) {

system("CLS");

cout << "\t\t\t-------------------------------------------\n";

cout << setw(20) << left << "" << "\t\t\tADD COMMIT" << endl;

cout << "\t\t\t-------------------------------------------\n";

string repoName, commitMessage;

cout << "\n\t\t\tEnter repository name: ";

cin >> repoName;

Repository\* repo = repositoryTree.searchRepository(repoName);

if (repo) {

cout << "\t\t\tEnter commit message: ";

cin.ignore();

getline(cin, commitMessage);

Commit\* newCommit = new Commit(commitMessage);

if (repo->addCommit(newCommit)) {

cout << "\n\t\t\tCOMMIT ADDED!" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

else {

cout << "\n\t\t\tFAILED TO ADD COMMIT! TRY AGAIN" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

delete newCommit;

}

}

else {

cout << "\n\t\t\tREPOSITORY NOT FOUND!" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

};

}

else if (choice == 8) {

system("CLS");

cout << "\t\t\t-------------------------------------------\n";

cout << setw(20) << left << "" << "\t\t\tADD FILE" << endl;

cout << "\t\t\t-------------------------------------------\n";

string repoName, fileName;

cout << "\n\t\t\tEnter repository name: ";

cin >> repoName;

Repository\* repo = repositoryTree.searchRepository(repoName);

if (repo) {

cout << "\t\t\tEnter file name: ";

cin >> fileName;

File\* newFile = new File(fileName);

if (repo->addFile(newFile)) {

cout << "\n\t\t\tFILE ADDED!" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

else {

cout << "\n\t\t\tFAILED TO ADD FILE!" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

delete newFile;

}

}

else {

cout << "\n\t\t\tREPOSITORY NOT FOUND!" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

}

else if (choice == 9) {

system("CLS");

cout << "\t\t\t-------------------------------------------\n";

cout << setw(20) << left << "" << "\t\t\tYOUR REPOSITORY" << endl;

cout << "\t\t\t-------------------------------------------\n";

string repoName;

cout << "\n\t\t\tEnter repository name: ";

cin >> repoName;

Repository\* repo = repositoryTree.searchRepository(repoName);

if (repo) {

cout << "\n\t\t\tRepository Name: " << repo->getName() << endl;

cout << "\n\t\t\tCommits:" << endl;

const Commit\*\* commits = repo->getCommits();

for (int i = 0; i < repo->getCommitCount(); ++i) {

cout << "\t\t\t- " << commits[i]->getMessage() << endl;

}

cout << "\n\t\t\tFiles:" << endl;

const File\*\* files = repo->getFiles();

for (int i = 0; i < repo->getFileCount(); ++i) {

cout << "\t\t\t- " << files[i]->getName() << endl << endl;

}

cout << "\t\t\t-------------------------------------------\n";

}

else {

cout << "\n\t\t\tREPOSITORY NOT FOUND!" << endl << endl;

cout << "\t\t\t-------------------------------------------\n";

}

}

else if (choice == 10) {

cout << "\t\t\tEXITING" << endl << endl;

repositoryTree.SaveData();

socialGraph.SaveData();

userManager.SaveData();

system("PAUSE");

return 0;

}

else {

cout << "\t\t\tINVALID OPTION! TRY AGAIN" << endl << endl;

}

system("PAUSE");

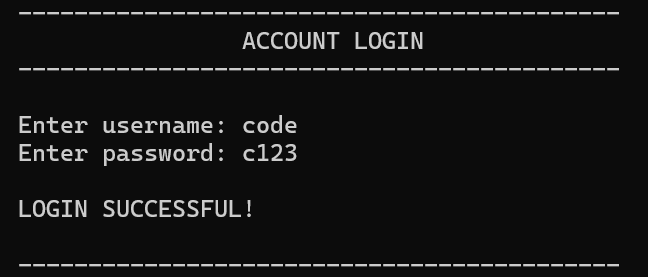
}

}

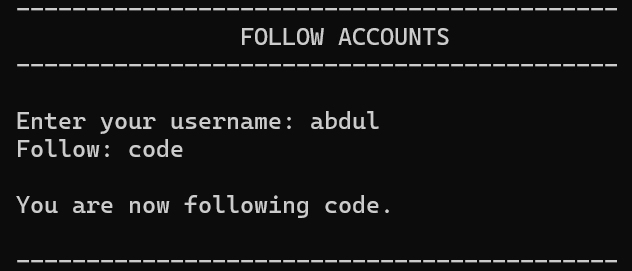
**Outputs:**

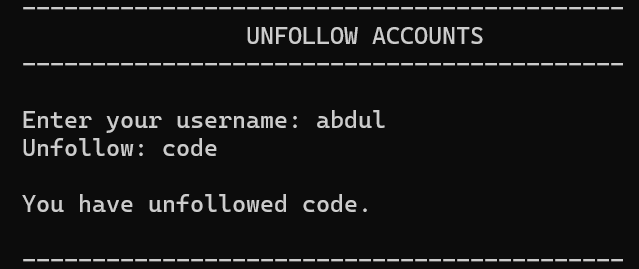
****

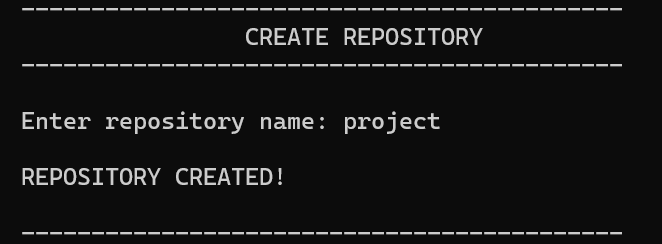
****

****

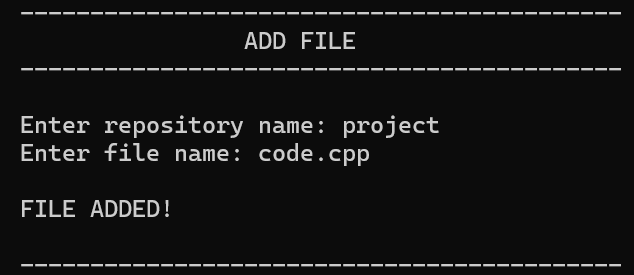
****

****

****

****

****

****

****