//Simple C++ Login System

#include <algorithm>

#include <fstream>

#include <iostream>

#include <sstream>

#include <string>

#include <vector>

using std::cin;

using std::cout;

using std::endl;

using std::getline;

using std::getline;

using std::string;

using std::stringstream;

using std::vector;

const char FIELD\_SEP = '|';

const string STUDENTS\_FILE = "students.txt";

const string ADMINS\_FILE = "admins.txt";

struct Student {

string username;

string password;

string fullName;

string studentId;

};

struct Admin {

string username;

string password;

string fullName;

};

static vector<Student> students;

static vector<Admin> admins;

/\* Utility helpers \*/

static string trim(const string &s) {

size\_t a = s.find\_first\_not\_of(" \t\r\n");

if (a == string::npos) return "";

size\_t b = s.find\_last\_not\_of(" \t\r\n");

return s.substr(a, b - a + 1);

}

static vector<string> splitFields(const string &line, char sep = FIELD\_SEP) {

vector<string> fields;

string cur;

std::istringstream ss(line);

while (std::getline(ss, cur, sep)) {

fields.push\_back(cur);

}

return fields;

}

/\* Persistence \*/

void saveStudentsToFile() {

std::ofstream out(STUDENTS\_FILE, std::ios::trunc);

if (!out) {

cout << "Failed to open " << STUDENTS\_FILE << " for writing.\n";

return;

}

for (const auto &s : students) {

out << s.username << FIELD\_SEP

<< s.password << FIELD\_SEP

<< s.fullName << FIELD\_SEP

<< s.studentId << "\n";

}

out.close();

}

void saveAdminsToFile() {

std::ofstream out(ADMINS\_FILE, std::ios::trunc);

if (!out) {

cout << "Failed to open " << ADMINS\_FILE << " for writing.\n";

return;

}

for (const auto &a : admins) {

out << a.username << FIELD\_SEP

<< a.password << FIELD\_SEP

<< a.fullName << "\n";

}

out.close();

}

void loadStudentsFromFile() {

students.clear();

std::ifstream in(STUDENTS\_FILE);

if (!in) {

// File absent => start with empty list

return;

}

string line;

while (std::getline(in, line)) {

if (trim(line).empty()) continue;

auto f = splitFields(line);

if (f.size() >= 4) {

Student s;

s.username = f[0];

s.password = f[1];

s.fullName = f[2];

s.studentId = f[3];

students.push\_back(s);

}

}

in.close();

}

void loadAdminsFromFile() {

admins.clear();

std::ifstream in(ADMINS\_FILE);

if (!in) {

// File absent => create default admin

Admin a;

a.username = "admin";

a.password = "admin123";

a.fullName = "Default Administrator";

admins.push\_back(a);

saveAdminsToFile();

return;

}

string line;

while (std::getline(in, line)) {

if (trim(line).empty()) continue;

auto f = splitFields(line);

if (f.size() >= 3) {

Admin a;

a.username = f[0];

a.password = f[1];

a.fullName = f[2];

admins.push\_back(a);

}

}

in.close();

if (admins.empty()) {

// ensure at least one admin

Admin a;

a.username = "admin";

a.password = "admin123";

a.fullName = "Default Administrator";

admins.push\_back(a);

saveAdminsToFile();

}

}

/\* Lookup helpers \*/

Student\* findStudentByUsername(const string &username) {

for (auto &s : students) {

if (s.username == username) return &s;

}

return nullptr;

}

Admin\* findAdminByUsername(const string &username) {

for (auto &a : admins) {

if (a.username == username) return &a;

}

return nullptr;

}

/\* Registration & Login \*/

void registerStudent() {

cout << "\n--- Student Registration ---\n";

string username;

cout << "Choose a username: ";

getline(cin, username);

username = trim(username);

if (username.empty()) {

cout << "Username cannot be empty.\n";

return;

}

if (findStudentByUsername(username) != nullptr) {

cout << "That username is already taken.\n";

return;

}

string fullName;

cout << "Full name: ";

getline(cin, fullName);

fullName = trim(fullName);

string studentId;

cout << "Student ID: ";

getline(cin, studentId);

studentId = trim(studentId);

string password;

cout << "Password: ";

getline(cin, password);

password = trim(password);

if (password.empty()) {

cout << "Password cannot be empty.\n";

return;

}

Student s;

s.username = username;

s.fullName = fullName;

s.studentId = studentId;

s.password = password;

students.push\_back(s);

saveStudentsToFile();

cout << "Registration successful! You can now log in as '" << username << "'.\n";

}

Student\* studentLogin() {

cout << "\n--- Student Login ---\n";

string username;

cout << "Username: ";

getline(cin, username);

username = trim(username);

string password;

cout << "Password: ";

getline(cin, password);

password = trim(password);

Student\* s = findStudentByUsername(username);

if (s == nullptr) {

cout << "No such student username.\n";

return nullptr;

}

if (s->password != password) {

cout << "Incorrect password.\n";

return nullptr;

}

cout << "Login successful. Welcome, " << s->fullName << "!\n";

return s;

}

Admin\* adminLogin() {

cout << "\n--- Admin Login ---\n";

string username;

cout << "Admin username: ";

getline(cin, username);

username = trim(username);

string password;

cout << "Password: ";

getline(cin, password);

password = trim(password);

Admin\* a = findAdminByUsername(username);

if (a == nullptr) {

cout << "No such admin username.\n";

return nullptr;

}

if (a->password != password) {

cout << "Incorrect password.\n";

return nullptr;

}

cout << "Admin login successful. Hello, " << a->fullName << "!\n";

return a;

}

/\* Student menu \*/

void studentMenu(Student &me) {

while (true) {

cout << "\n--- Student Menu (" << me.username << ") ---\n";

cout << "1) View profile\n";

cout << "2) Change password\n";

cout << "3) Logout\n";

cout << "Select: ";

string choice;

getline(cin, choice);

if (choice == "1") {

cout << "\n--- Profile ---\n";

cout << "Username: " << me.username << "\n";

cout << "Full name: " << me.fullName << "\n";

cout << "Student ID: " << me.studentId << "\n";

} else if (choice == "2") {

cout << "Enter current password: ";

string cur;

getline(cin, cur);

if (cur != me.password) {

cout << "Incorrect current password.\n";

} else {

cout << "Enter new password: ";

string np;

getline(cin, np);

if (np.empty()) {

cout << "Password cannot be empty.\n";

} else {

me.password = np;

saveStudentsToFile();

cout << "Password changed.\n";

}

}

} else if (choice == "3") {

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

break;

} else {

cout << "Invalid choice.\n";

}

}

}

/\* Admin actions \*/

void listAllStudents() {

cout << "\n--- Students List ---\n";

if (students.empty()) {

cout << "(no students)\n";

return;

}

for (size\_t i = 0; i < students.size(); ++i) {

const auto &s = students[i];

cout << i + 1 << ") " << s.username << " - " << s.fullName << " - ID: " << s.studentId << "\n";

}

}

void adminAddStudent() {

cout << "\n--- Admin: Add Student ---\n";

string username;

cout << "Username: ";

getline(cin, username);

username = trim(username);

if (username.empty()) {

cout << "Username cannot be empty.\n";

return;

}

if (findStudentByUsername(username) != nullptr) {

cout << "That username is already taken.\n";

return;

}

string fullName;

cout << "Full name: ";

getline(cin, fullName);

string studentId;

cout << "Student ID: ";

getline(cin, studentId);

string password;

cout << "Initial password: ";

getline(cin, password);

Student s;

s.username = username;

s.fullName = fullName;

s.studentId = studentId;

s.password = password;

students.push\_back(s);

saveStudentsToFile();

cout << "Student added.\n";

}

void adminRemoveStudent() {

cout << "\n--- Admin: Remove Student ---\n";

string username;

cout << "Username to remove: ";

getline(cin, username);

username = trim(username);

auto it = std::find\_if(students.begin(), students.end(),

[&](const Student &s) { return s.username == username; });

if (it == students.end()) {

cout << "No such student.\n";

return;

}

cout << "Are you sure you want to remove student '" << it->username << "'? (y/N): ";

string ans;

getline(cin, ans);

if (!ans.empty() && (ans[0] == 'y' || ans[0] == 'Y')) {

students.erase(it);

saveStudentsToFile();

cout << "Student removed.\n";

} else {

cout << "Cancelled.\n";

}

}

void adminResetStudentPassword() {

cout << "\n--- Admin: Reset Student Password ---\n";

string username;

cout << "Username: ";

getline(cin, username);

username = trim(username);

Student\* s = findStudentByUsername(username);

if (!s) {

cout << "No such student.\n";

return;

}

cout << "New password: ";

string np;

getline(cin, np);

if (np.empty()) {

cout << "Password cannot be empty.\n";

return;

}

s->password = np;

saveStudentsToFile();

cout << "Password reset for " << s->username << ".\n";

}

void adminMenu(Admin &me) {

while (true) {

cout << "\n--- Admin Menu (" << me.username << ") ---\n";

cout << "1) List students\n";

cout << "2) Add student\n";

cout << "3) Remove student\n";

cout << "4) Reset student password\n";

cout << "5) Change my password\n";

cout << "6) Logout\n";

cout << "Select: ";

string choice;

getline(cin, choice);

if (choice == "1") {

listAllStudents();

} else if (choice == "2") {

adminAddStudent();

} else if (choice == "3") {

adminRemoveStudent();

} else if (choice == "4") {

adminResetStudentPassword();

} else if (choice == "5") {

cout << "Enter current password: ";

string cur;

getline(cin, cur);

if (cur != me.password) {

cout << "Incorrect current password.\n";

} else {

cout << "Enter new password: ";

string np;

getline(cin, np);

if (np.empty()) {

cout << "Password cannot be empty.\n";

} else {

me.password = np;

saveAdminsToFile();

cout << "Password changed.\n";

}

}

} else if (choice == "6") {

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

break;

} else {

cout << "Invalid choice.\n";

}

}

}

/\* Main menu and program flow \*/

void mainMenu() {

while (true) {

cout << "\n=== Welcome to the Login System ===\n";

cout << "1) Student login\n";

cout << "2) Student registration\n";

cout << "3) Admin login\n";

cout << "4) Quit\n";

cout << "Select: ";

string choice;

getline(cin, choice);

if (choice == "1") {

Student\* s = studentLogin();

if (s) studentMenu(\*s);

} else if (choice == "2") {

registerStudent();

} else if (choice == "3") {

Admin\* a = adminLogin();

if (a) adminMenu(\*a);

} else if (choice == "4") {

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

break;

} else {

cout << "Invalid selection.\n";

}

}

}

int main() {

loadAdminsFromFile();

loadStudentsFromFile();

// Add a demo student if none exist, so the program is easy to try.

if (students.empty()) {

Student demo;

demo.username = "student1";

demo.password = "pass123";

demo.fullName = "Demo Student";

demo.studentId = "S1001";

students.push\_back(demo);

saveStudentsToFile();

}

mainMenu();

return 0;

}