#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <iomanip>

using namespace std;

const string ClientsFileName = "Clients.txt";

struct sClient {

string AccountNumber;

string PinCode;

string Name;

string Phone;

double AccountBalance;

bool MarkForDelete = false;

};

vector<string> SplitString(string str, string delim) {

vector<string> result;

size\_t pos = 0;

string token;

while ((pos = str.find(delim)) != string::npos) {

token = str.substr(0, pos);

if (!token.empty()) result.push\_back(token);

str.erase(0, pos + delim.length());

}

if (!str.empty()) result.push\_back(str);

return result;

}

sClient ConvertLineToClient(string line, string sep = "#//#") {

vector<string> data = SplitString(line, sep);

sClient client;

if (data.size() == 5) {

client.AccountNumber = data[0];

client.PinCode = data[1];

client.Name = data[2];

client.Phone = data[3];

client.AccountBalance = stod(data[4]);

}

return client;

}

string ConvertClientToLine(sClient client, string sep = "#//#") {

return client.AccountNumber + sep + client.PinCode + sep + client.Name + sep + client.Phone + sep + to\_string(client.AccountBalance);

}

vector<sClient> LoadClients(string filename) {

vector<sClient> clients;

ifstream file(filename);

string line;

while (getline(file, line)) {

sClient client = ConvertLineToClient(line);

clients.push\_back(client);

}

file.close();

return clients;

}

void SaveClients(string filename, vector<sClient> clients) {

ofstream file(filename);

for (const sClient& client : clients) {

if (!client.MarkForDelete)

file << ConvertClientToLine(client) << endl;

}

file.close();

}

void ShowAllClients(vector<sClient> clients) {

cout << "\nClient List (" << clients.size() << "):\n";

cout << left << setw(15) << "Account" << setw(10) << "PIN" << setw(25) << "Name" << setw(15) << "Phone" << "Balance\n";

for (const sClient& client : clients) {

cout << left << setw(15) << client.AccountNumber << setw(10) << client.PinCode

<< setw(25) << client.Name << setw(15) << client.Phone << client.AccountBalance << endl;

}

}

void AddNewClient(vector<sClient>& clients) {

sClient client;

cout << "Enter Account Number: ";

cin >> client.AccountNumber;

cout << "Enter PIN: ";

cin >> client.PinCode;

cin.ignore();

cout << "Enter Name: ";

getline(cin, client.Name);

cout << "Enter Phone: ";

getline(cin, client.Phone);

cout << "Enter Balance: ";

cin >> client.AccountBalance;

clients.push\_back(client);

SaveClients(ClientsFileName, clients);

cout << "Client added successfully.\n";

}

void DeleteClient(vector<sClient>& clients) {

string accNo;

cout << "Enter Account Number to delete: ";

cin >> accNo;

for (sClient& client : clients) {

if (client.AccountNumber == accNo) {

client.MarkForDelete = true;

SaveClients(ClientsFileName, clients);

cout << "Client deleted.\n";

return;

}

}

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

}

void MainMenu() {

vector<sClient> clients = LoadClients(ClientsFileName);

int choice;

do {

cout << "\n--- Main Menu ---\n";

cout << "1. Show Clients\n2. Add Client\n3. Delete Client\n4. Exit\nChoice: ";

cin >> choice;

switch (choice) {

case 1: ShowAllClients(clients); break;

case 2: AddNewClient(clients); clients = LoadClients(ClientsFileName); break;

case 3: DeleteClient(clients); clients = LoadClients(ClientsFileName); break;

case 4: cout << "Exiting...\n"; break;

default: cout << "Invalid option.\n";

}

} while (choice != 4);

}

int main() {

MainMenu();

return 0;

}