#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

using namespace std;

template <typename T1, typename T2>

struct Appointment {

T1 id;

string name;

string type;

int timeUntil;

T2 bill;

string status;

};

template <typename T1, typename T2>

int loadAppointments(Appointment<T1, T2> appointments[]);

template <typename T1, typename T2>

void saveAppointments(Appointment<T1, T2> appointments[], int count);

template <typename T1, typename T2>

void addAppointment(Appointment<T1, T2> appointments[], int& count);

template <typename T1, typename T2>

void displayAppointments(Appointment<T1, T2> appointments[], int count);

template <typename T1, typename T2>

void markCompleted(Appointment<T1, T2> appointments[], int count);

template <typename T1, typename T2>

void appointmentSummary(Appointment<T1, T2> appointments[], int count);

int main() {

Appointment<string, float> appointments[100];

int count = loadAppointments(appointments);

int choice;

do {

cout << "\n\*\*\*\*\*\*\* Welcome to Clinic Appointment Manager \*\*\*\*\*\*\*\n";

cout << "1. Add New Appointment\n";

cout << "2. Display All Appointments\n";

cout << "3. Mark Appointment as Completed\n";

cout << "4. View Appointment Summary\n";

cout << "5. Exit\n→ ";

cin >> choice;

switch (choice) {

case 1: addAppointment(appointments, count); break;

case 2: displayAppointments(appointments, count); break;

case 3: markCompleted(appointments, count); break;

case 4: appointmentSummary(appointments, count); break;

case 5:

saveAppointments(appointments, count);

cout << "Data saved to appointments.txt. Goodbye!" << endl;

break;

default:

cout << "Invalid option. Try again.\n";

}

} while (choice != 5);

return 0;

}

template <typename T1, typename T2>

int loadAppointments(Appointment<T1, T2> appointments[]) {

ifstream file("appointments.txt");

int count = 0;

if (!file) {

cout << "No existing file found. Starting with empty list.\n";

return 0;

}

while (!file.eof()) {

Appointment<T1, T2> temp;

char comma;

file >> ws;

getline(file, temp.id, ',');

if (temp.id.empty()) break;

getline(file, temp.name, ',');

getline(file, temp.type, ',');

file >> temp.timeUntil >> comma >> temp.bill >> comma;

getline(file, temp.status);

appointments[count++] = temp;

}

file.close();

return count;

}

template <typename T1, typename T2>

void saveAppointments(Appointment<T1, T2> appointments[], int count) {

ofstream file("appointments.txt");

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

file << appointments[i].id << ","

<< appointments[i].name << ","

<< appointments[i].type << ","

<< appointments[i].timeUntil << ","

<< fixed << setprecision(2) << appointments[i].bill << ","

<< appointments[i].status << "\n";

}

file.close();

}

template <typename T1, typename T2>

void addAppointment(Appointment<T1, T2> appointments[], int& count) {

Appointment<T1, T2> newApp;

cout << "Enter appointment ID: ";

cin >> ws;

getline(cin, newApp.id);

cout << "Enter patient name: ";

getline(cin, newApp.name);

cout << "Enter appointment type (General/Dental/Emergency/etc.): ";

getline(cin, newApp.type);

cout << "Enter time until appointment (hours): ";

cin >> newApp.timeUntil;

cout << "Enter estimated bill: ";

cin >> newApp.bill;

newApp.status = "Pending";

appointments[count++] = newApp;

cout << "Appointment added!\n";

}

template <typename T1, typename T2>

void displayAppointments(Appointment<T1, T2> appointments[], int count) {

if (count == 0) {

cout << "No appointments found.\n";

return;

}

cout << "\n=========== APPOINTMENTS ===========\n";

cout << left << setw(10) << "ID" << setw(15) << "Name"

<< setw(15) << "Type" << setw(10) << "Time"

<< setw(10) << "Bill" << setw(12) << "Status" << endl;

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

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

cout << left << setw(10) << appointments[i].id

<< setw(15) << appointments[i].name

<< setw(15) << appointments[i].type

<< setw(10) << appointments[i].timeUntil

<< "$" << setw(9) << fixed << setprecision(2) << appointments[i].bill

<< setw(12) << appointments[i].status << endl;

}

}

template <typename T1, typename T2>

void markCompleted(Appointment<T1, T2> appointments[], int count) {

if (count == 0) {

cout << "No appointments to update.\n";

return;

}

cout << "Enter appointment ID to mark as completed: ";

string targetID;

cin >> ws;

getline(cin, targetID);

bool found = false;

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

if (appointments[i].id == targetID) {

appointments[i].status = "Completed";

cout << "Appointment " << targetID << " marked as Completed.\n";

found = true;

break;

}

}

if (!found) {

cout << "Appointment ID not found.\n";

}

}

template <typename T1, typename T2>

void appointmentSummary(Appointment<T1, T2> appointments[], int count) {

int completed = 0;

int pending = 0;

double totalEarnings = 0.0;

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

if (appointments[i].status == "Completed") {

completed++;

totalEarnings += appointments[i].bill;

}

else {

pending++;

}

}

cout << "\n======== APPOINTMENT SUMMARY ========\n";

cout << "Total Appointments: " << count << endl;

cout << "Completed: " << completed << endl;

cout << "Pending: " << pending << endl;

cout << "Total Earnings: $" << fixed << setprecision(2) << totalEarnings << endl;

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

}