TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH

**KHOA CÔNG NGHỆ THÔNG TIN**

---🙡 🕮 🙣---



**BÁO CÁO ĐỀ TÀI MÔN HỌC**

ĐỀ TÀI: **KIỂM THỬ ỨNG DỤNG ĐẶT LỊCH CẮT TÓC**

|  |  |
| --- | --- |
| **Sinh viên thực hiện:** | **Tô Trọng Nhân** |
| **Mã số sinh viên:** | **2151013063** |
| **Lớp:** | **DH21CS01** |
| **Giảng viên hướng dẫn:** | **Th.s Nguyễn Thị Ngọc Thanh** |

**Tháng 01 năm 2024**

Table of Contents

[LỜI MỞ ĐẦU 3](#_Toc156204657)

[CHƯƠNG 1. ỨNG DỤNG ĐẶT LỊCH CẮT TÓC 4](#_Toc156204658)

[1.1CÁC CẤU HÌNH TRONG ỨNG DỤNG 4](#_Toc156204659)

[1.2 GIAO DIỆN CỦA ỨNG DỤNG 6](#_Toc156204660)

[CHƯƠNG 2. VIẾT CÁC UNIT TEST KIỂM THỬ 3 CHỨC NĂNG ĐĂNG NHẬP, ĐĂNG KÝ VÀ ĐẶT LỊCH 20](#_Toc156204661)

[2.1 Chuẩn bị 20](#_Toc156204662)

[2.2 Testcase: 24](#_Toc156204663)

[2.3 Kết luận 41](#_Toc156204664)

[CHƯƠNG 3. VIẾT CÁC TEST ĐỂ TEST CÁC APIs 42](#_Toc156204665)

[3.1 Setup API 42](#_Toc156204666)

[3.2 Test api 48](#_Toc156204667)

# LỜI MỞ ĐẦU

Ngày nay con người rất thích và rất chú trọng vào việc làm đẹp cho bản thân, đặc biệt là việc cắt tóc bởi vì người xưa thường có nói “Cái răng cái tóc là góc con người”. Bởi vì nắm được những nhu cầu đó của con người nên các barbers đã đưa ứng dụng Đặt Lịch Cắt Tóc vào các barbershop của họ để giúp mọi người có thể dễ dàng đặt lịch cắt tóc trước, tránh trường hợp phải chờ đợi khi có mặt ở cửa hàng

Kiểm thử ứng dụng Đặt Lịch Cắt Tóc không chỉ đảm bảo rằng các tính năng cơ bản như đăng ký, đăng nhập, chọn dịch vụ, đặt lịch và thanh toán hoạt động một cách đúng đắn, mà còn chú trọng đến khả năng mở rộng của hệ thống để đối mặt với sự phát triển của hệ thống sau này.

Các bước kiểm thử có thể bao gồm kiểm tra tính năng bảo mật để đảm bảo an toàn thông tin cá nhân và giao dịch thanh toán. Đồng thời, kiểm thử hiệu suất giúp đánh giá khả năng xử lý của hệ thống khi có nhiều người dùng truy cập cùng một lúc. Kiểm thử giao diện người dùng đóng vai trò quan trọng trong việc đảm bảo trải nghiệm người dùng mượt mà và dễ sử dụng. Việc này đặc biệt quan trọng khi người dùng thực hiện các hoạt động như đăng ký, đăng nhập, chọn dịch vụ, đặt lịch, thanh toán. Cuối cùng, kiểm thử phải được thực hiện không chỉ trong môi trường thử nghiệm, mà còn trên môi trường thực tế để đảm bảo rằng hệ thống hoạt động ổn định và đáp ứng mọi thách thức mà môi trường sản xuất có thể đưa ra.

## CHƯƠNG 1. ỨNG DỤNG ĐẶT LỊCH CẮT TÓC

### CÁC CẤU HÌNH TRONG ỨNG DỤNG

Sử dụng ngôn ngữ C#

IDE sử dụng: Visual Studio 2022

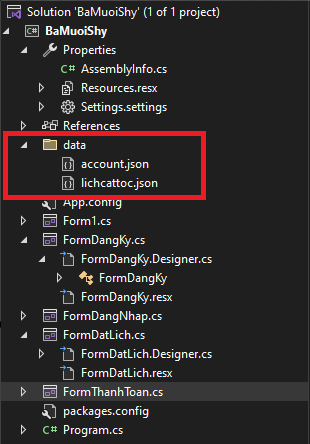
Hệ điều hành: Windows 11

Project: Windows Forms App(.NET framework)

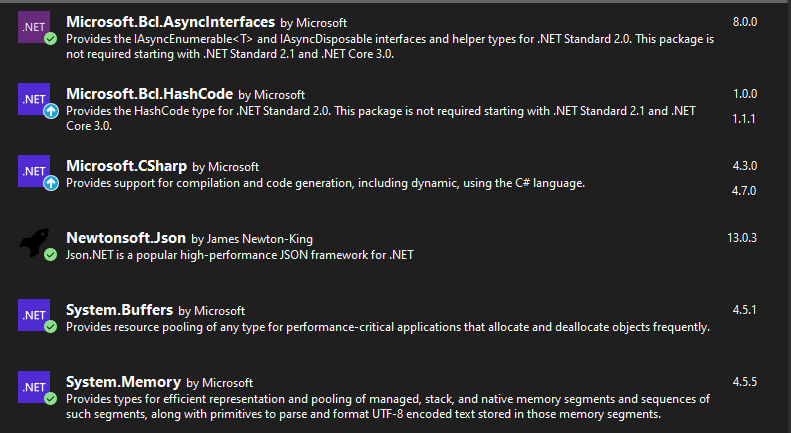
Target framework: .NET framework 4.8

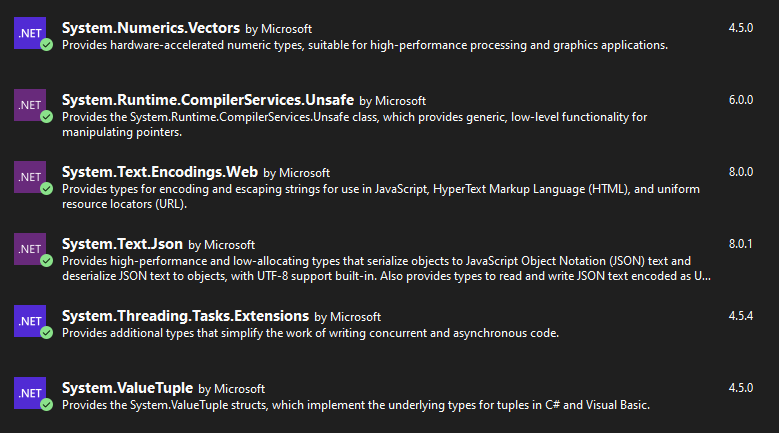
Dữ liệu: Account và Lịch Đặt Cắt Tóc được lưu trữ trong file 2 file JSON account.json và lichcattoc.json

2 file json được lưu trong folder data đặt trong project



Các thư viện sử dụng:



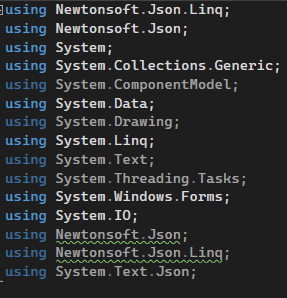


## 1.2 GIAO DIỆN CỦA ỨNG DỤNG

#### 1.2.1 Giao diện Đăng Ký

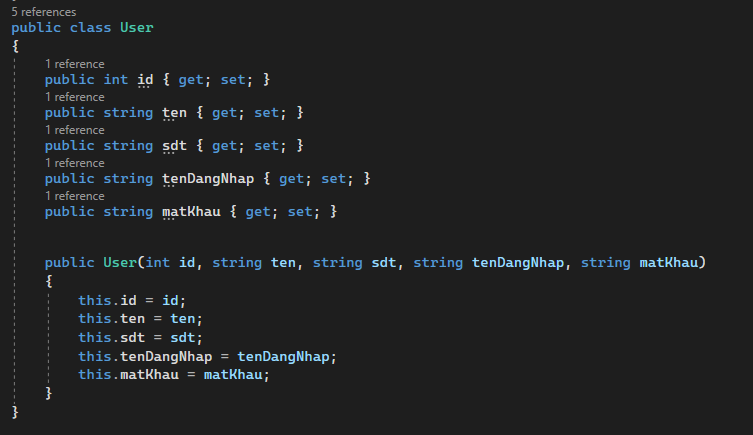


##### 1.2.1.1 Các thư viện được import

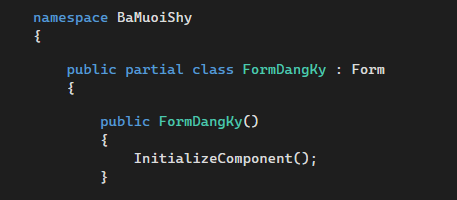


##### 1.2.1.2 Source code

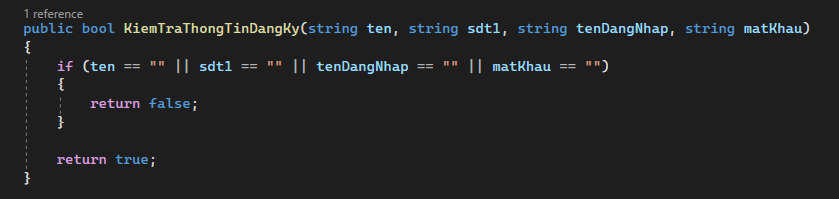
Tạo lớp đối tượng User để dễ dàng thao tác thêm các đối tượng vào file JSON

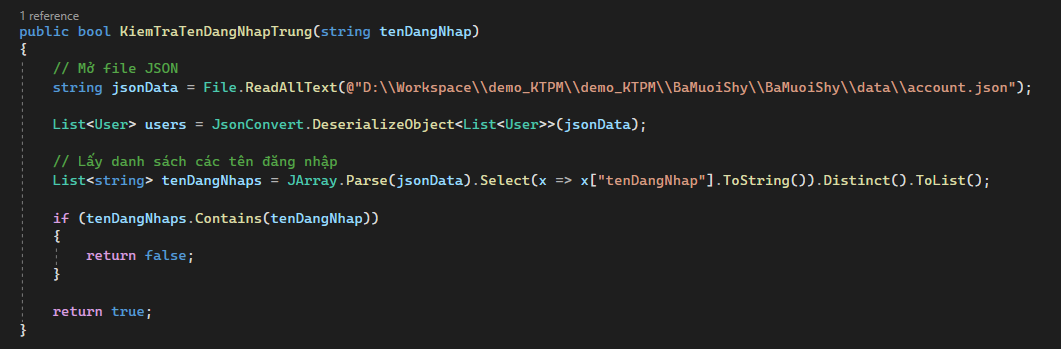


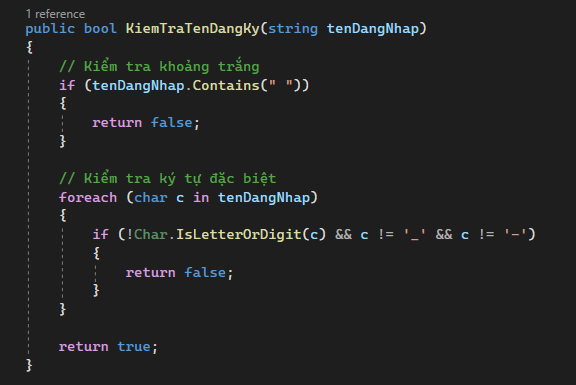
Khởi tạo form đăng ký



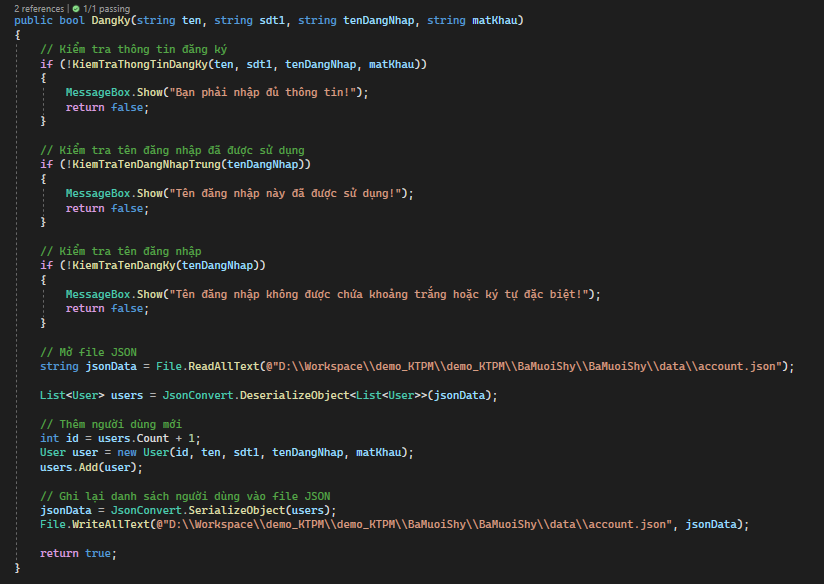
Hàm kiểm tra nhập đầy đủ thông tin khi đăng ký

****

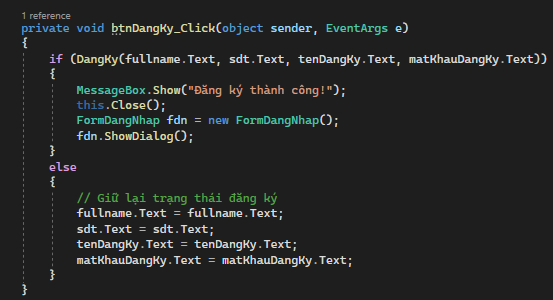
Hàm kiểm tra tên đăng nhập đã tồn tại hay chưa  


Hàm kiểm tra tên đăng nhập có ký tự đặt biệt hay có khoản trắng không  


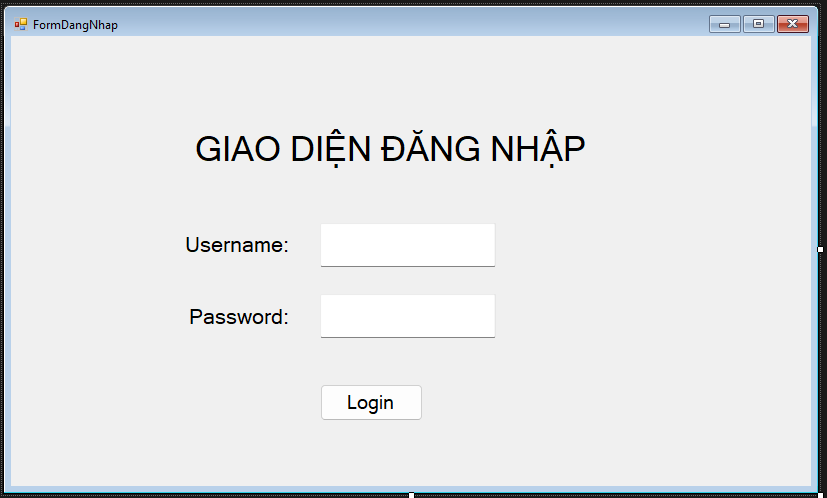
Hàm kiểm tra đăng ký



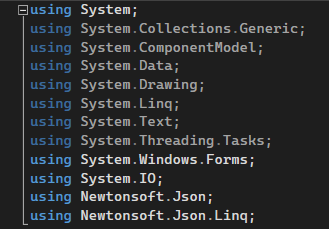
Button xử lý đăng ký



#### 1.2.2 Giao diện Đăng Nhập

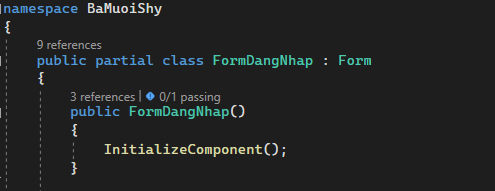


##### 1.2.2.1 Các thư viện được import

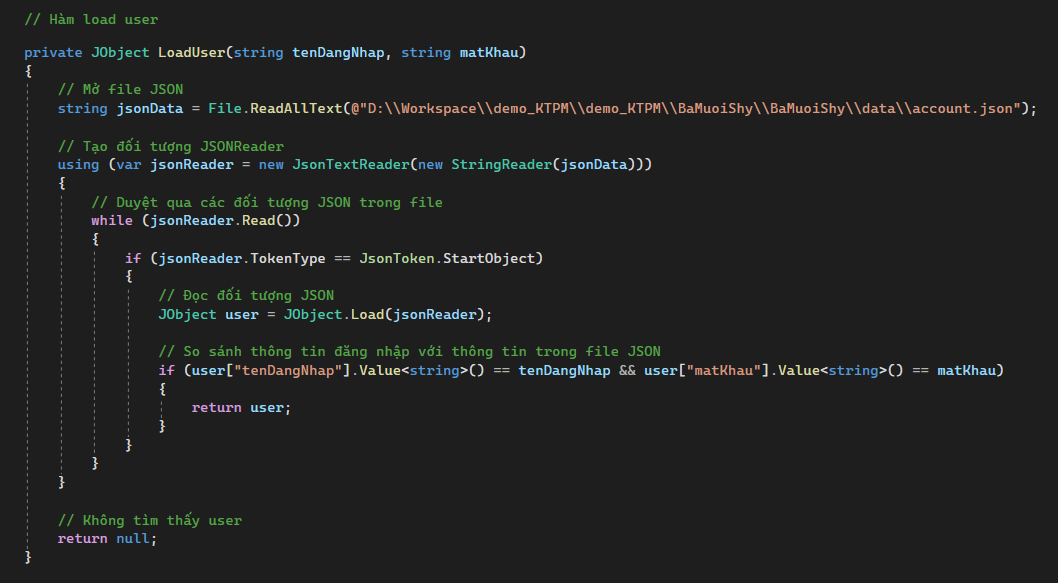


##### 1.2.2.2 Source code

Khởi tạo form

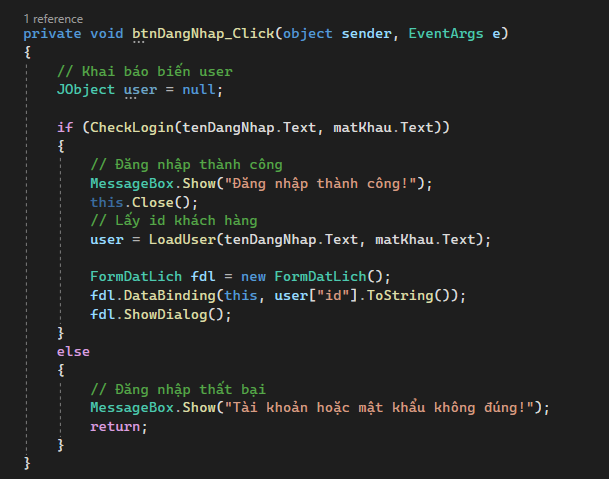


Hàm load user



Hàm kiểm tra đăng nhập  

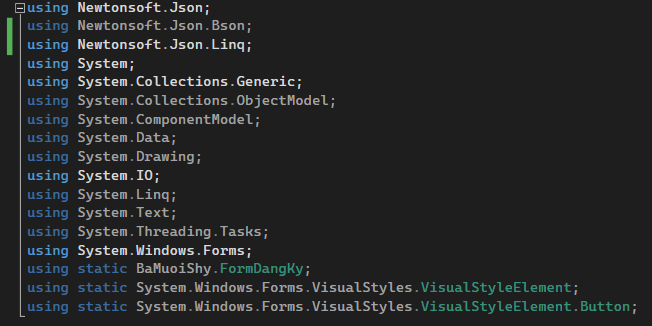

Xử lý button đăng nhập



#### 1.2.3 Giao diện Đặt Lịch

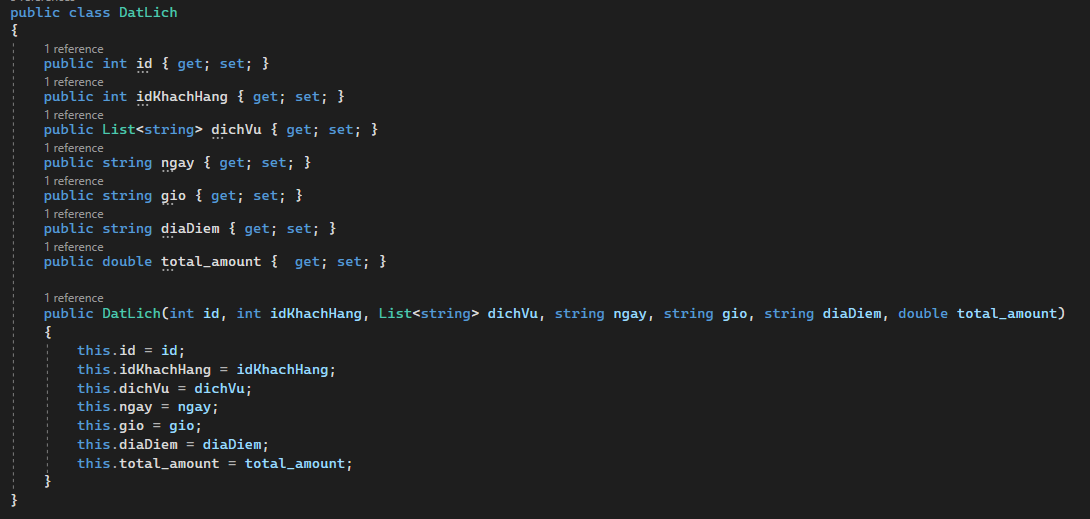


##### 1.2.3.1 Các thư viện được import

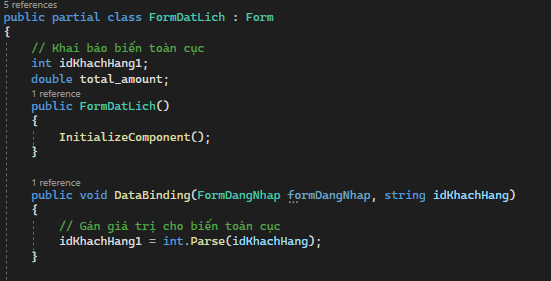


##### 1.2.3.2 Source code

Tạo lớp đối tượng DatLich cho dễ dàng thao tác với file JSON

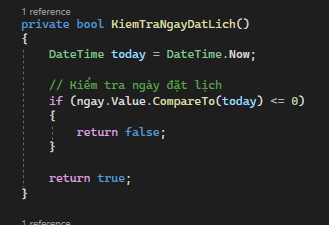


Khởi tạo form đặt lịch

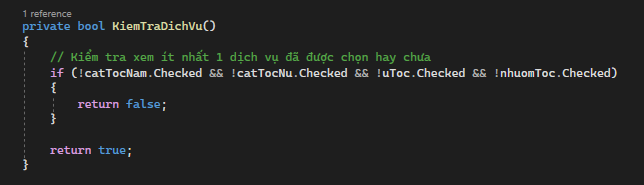


Có 2 biến toàn cục là idKhachHang1 và total\_amount dùng để ghi nhớ id account đang đặt lịch, còn biến còn lại dùng để lưu tổng tiền cần phải thanh toán và để gửi qua cho Form thanh toán để hiển thị tổng tiền lên

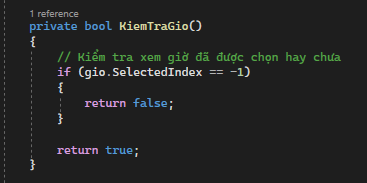
Hàm kiểm tra ngày đặt lịch



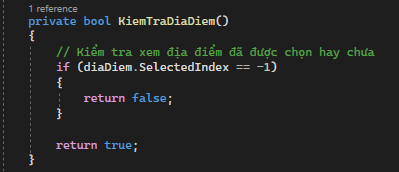
Hàm kiểm tra dịch vụ



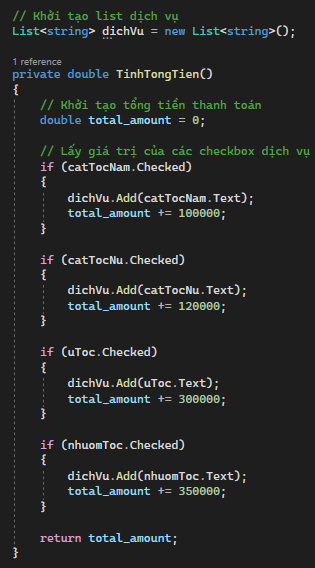
Hàm kiểm tra giờ đặt



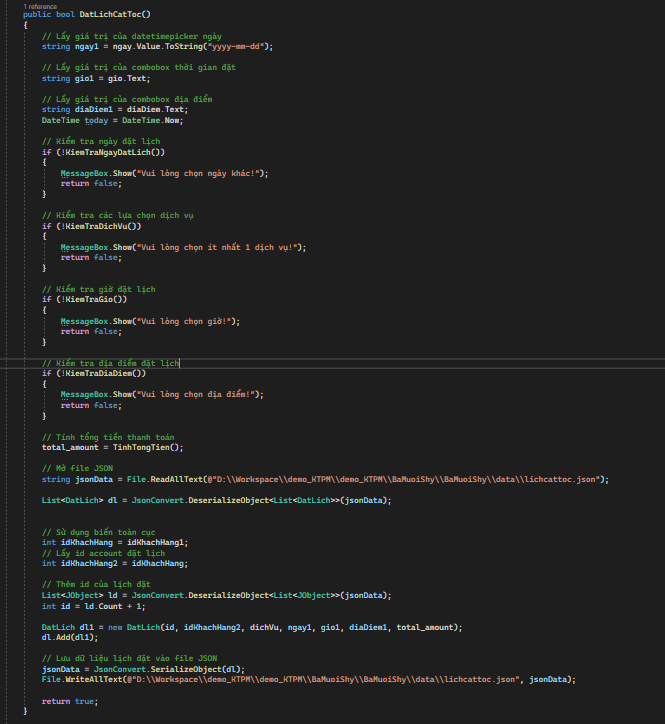
Hàm kiểm tra chọn địa điểm



Hàm tính tổng tiền và thêm những dịch vụ tick chọn vào biến toàn cục dichVu



Hàm đặt lịch cắt tóc



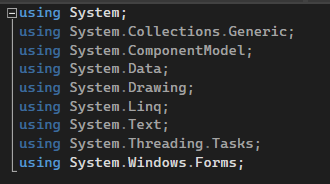
Button xử lý đặt lịch



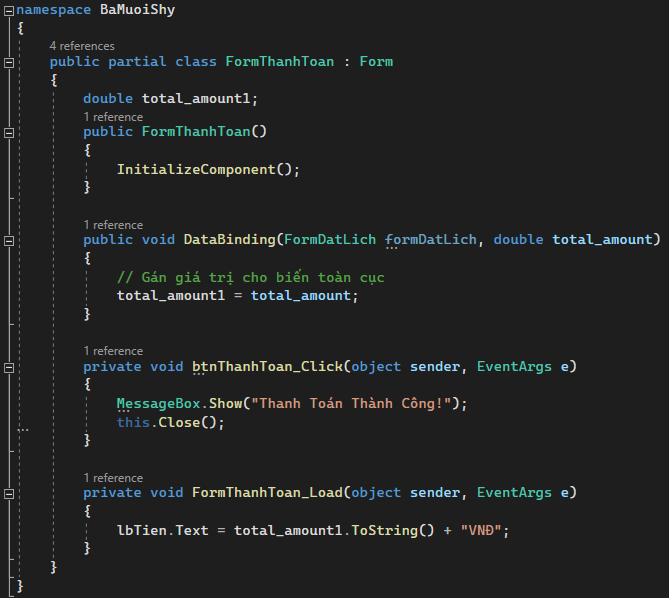
#### 1.2.4 Giao diện Thanh Toán



##### 1.2.4.1 Các thư viện được import



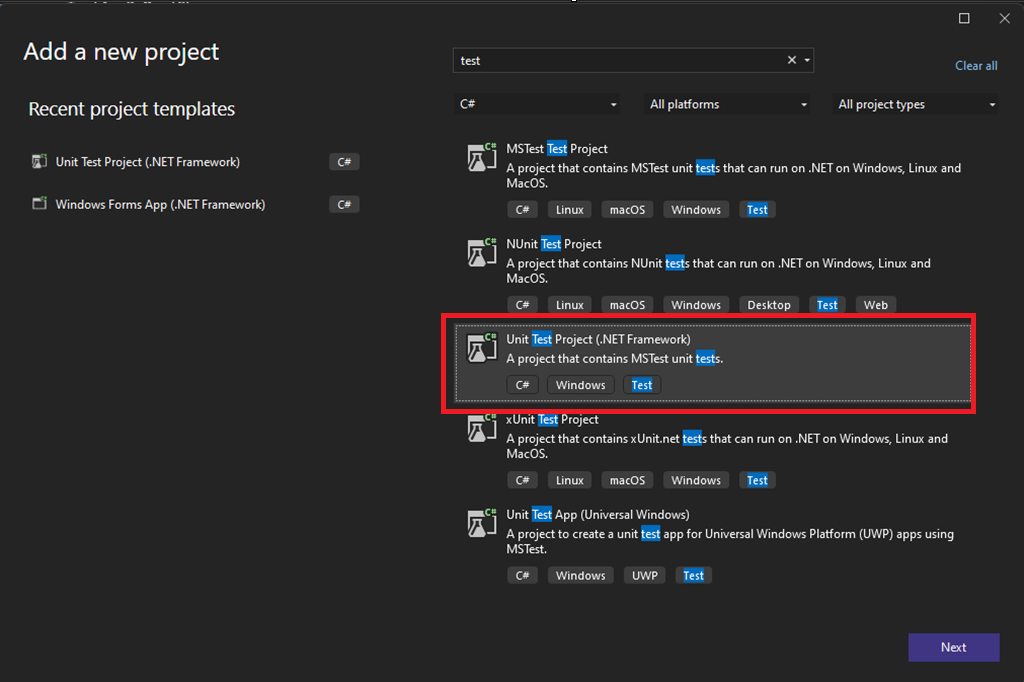
##### 1.2.4.1 Source code



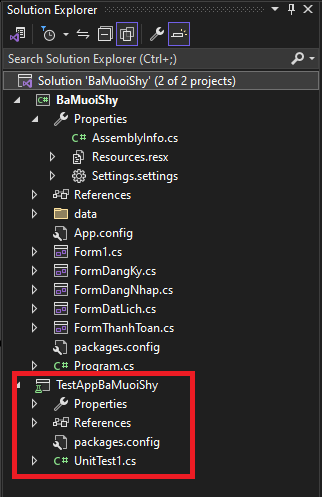
## CHƯƠNG 2. VIẾT CÁC UNIT TEST KIỂM THỬ 3 CHỨC NĂNG ĐĂNG NHẬP, ĐĂNG KÝ VÀ ĐẶT LỊCH

### 2.1 Chuẩn bị

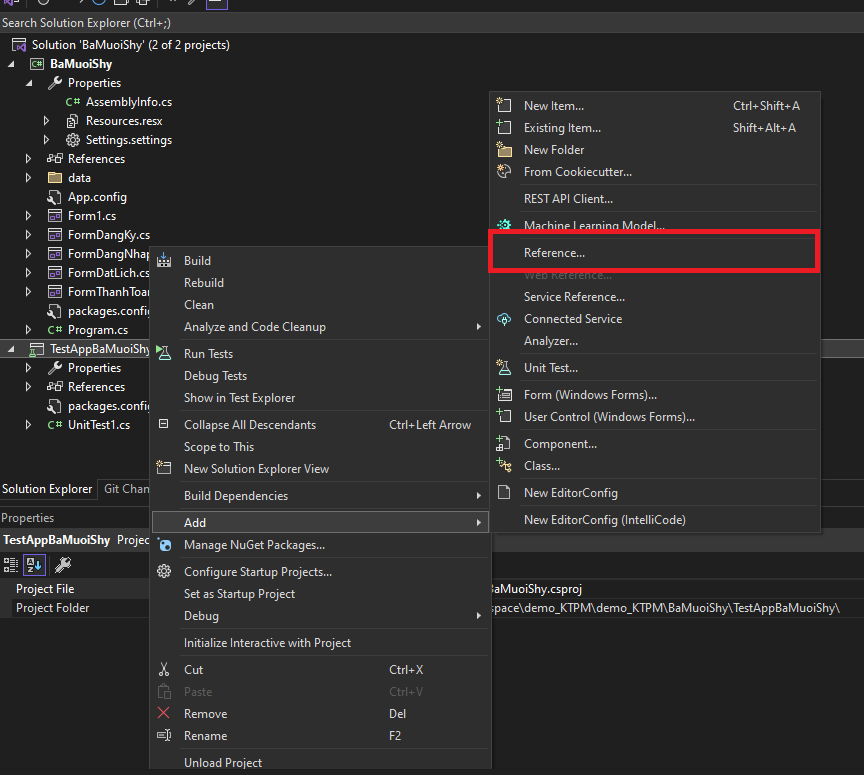
Bước đầu từ solution->add->new project, ta tạo 1 Unit Test Project(.NET)

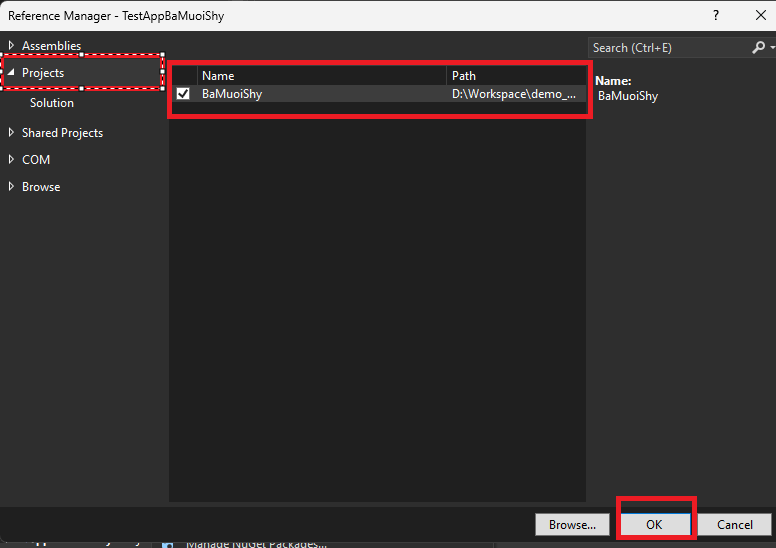


Đặt tên và ta được 1 project như này

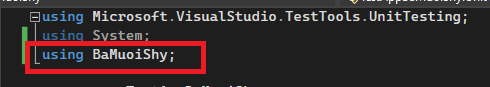


Sau đó ta add reference cho project test



Chọn sang project rồi add project đang làm vào  


Thêm project BaMuoiShy là project đặt lịch cắt tóc vào project test



### 2.2 Testcase:

#### 2.2.1 CN1\_Login:

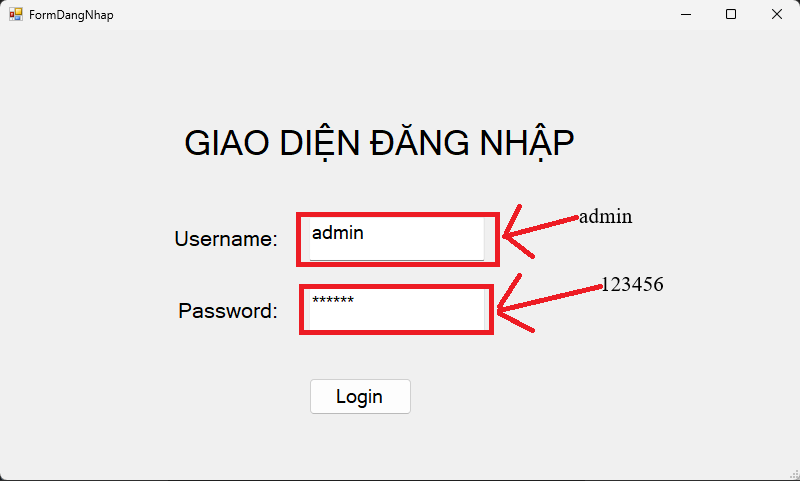
##### 2.2.1.1\_CheckLoginSuccess\_TC1

Đặc tả:

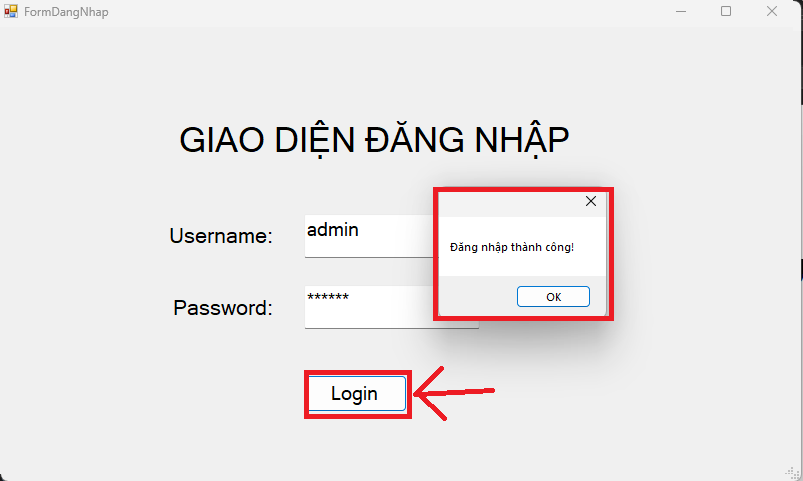
Mục đích: Kiểm tra đăng nhập thành công

Tiền điều kiện: Người dùng đã đăng ký và có tài khoản hợp lệ

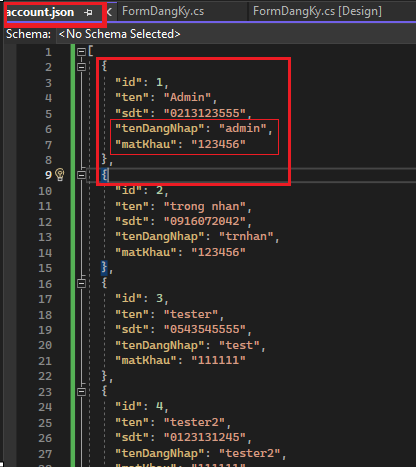
Kết quả mong đợi: Nếu tên đăng nhập và mật khẩu chính xác thì người dùng đăng nhập thành công



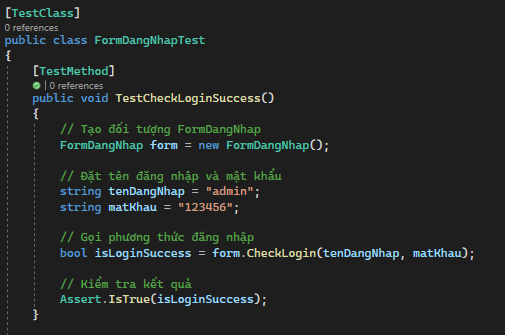
Bấm Login thì hiện thông báo đăng nhập thành công



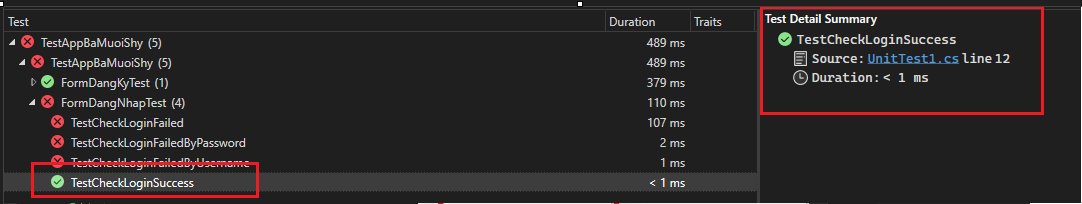
Đúng là vì account này đã có trong file account.json



Viết testcase



Chạy testcase



##### 2.2.1.2\_CheckLoginFailed\_TC2

Mục đích: Kiểm tra đăng nhập thất bại

Tiền điều kiện: Tên đăng nhập hoặc mật khẩu không chính xác

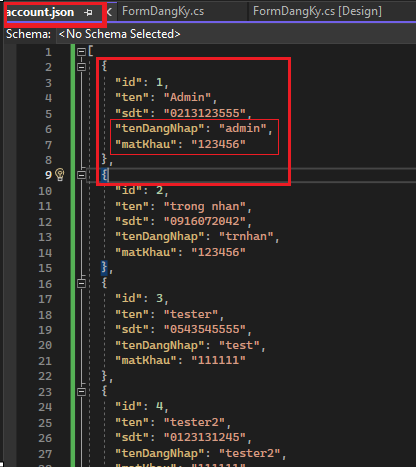
Kết quả mong đợi: Nếu tên đăng nhập và mật khẩu chính xác thì người dùng đăng nhập thành công



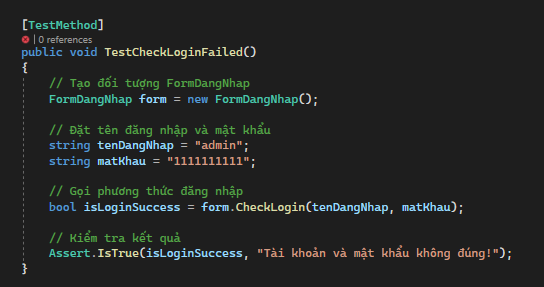
Bấm nút Login thì hiện thông báo đăng nhập failed



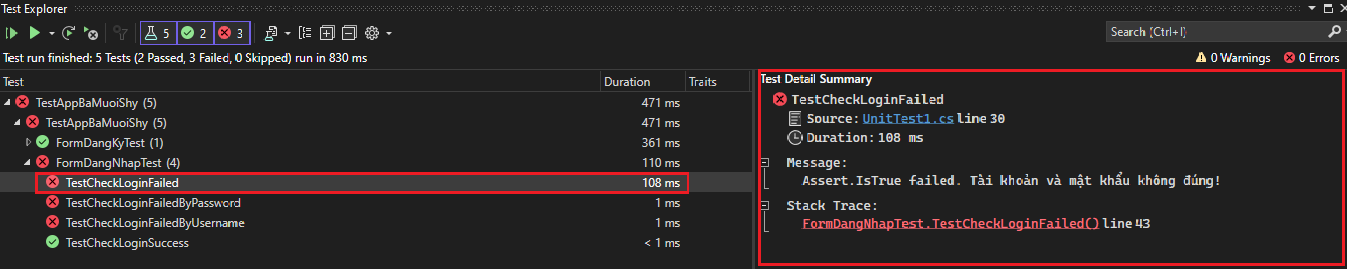
Không thể đăng nhập vì mật khẩu sai



Viết testcase



Chạy test



##### 2.2.1.3\_CheckLoginFailedByNoneUsername\_TC3

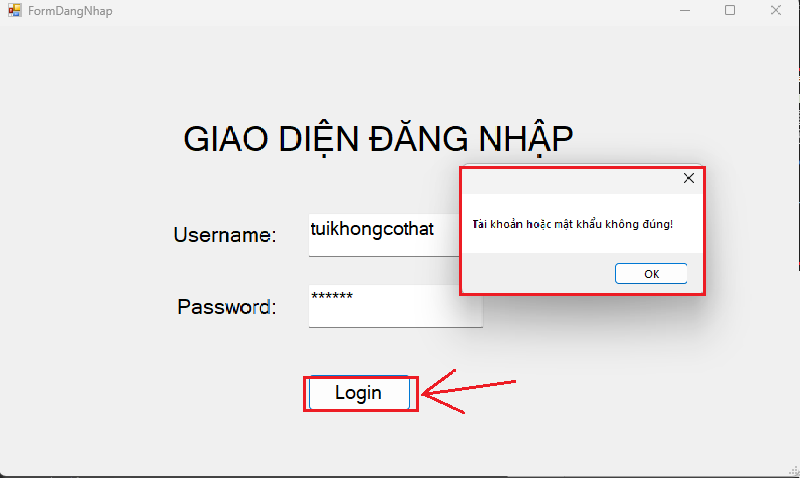
Mục đích: Kiểm tra đăng nhập với tài khoản không tồn tại

Tiền điều kiện: Tài khoản đăng nhập không tồn tại

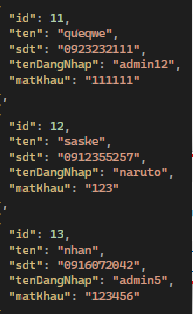
Kết quả mong đợi: Nếu tài khoản đăng nhập không tồn tại, người dùng vẫn sẽ nhận thông báo tài khoản hoặc mật khẩu sai để đảm bảo bảo mật về account người dùng



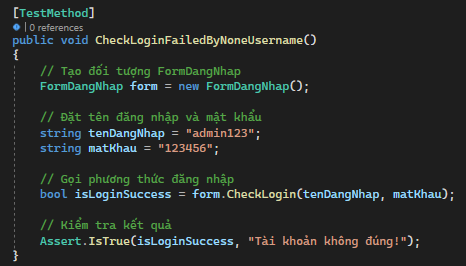
Bấm login vẫn sẽ hiện thông báo giống khi failed



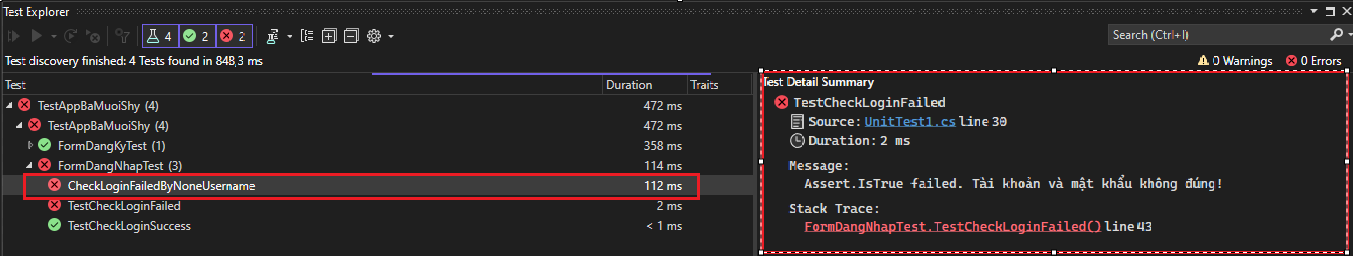
Trong account.json không có username này



Viết testcase



Chạy testcase



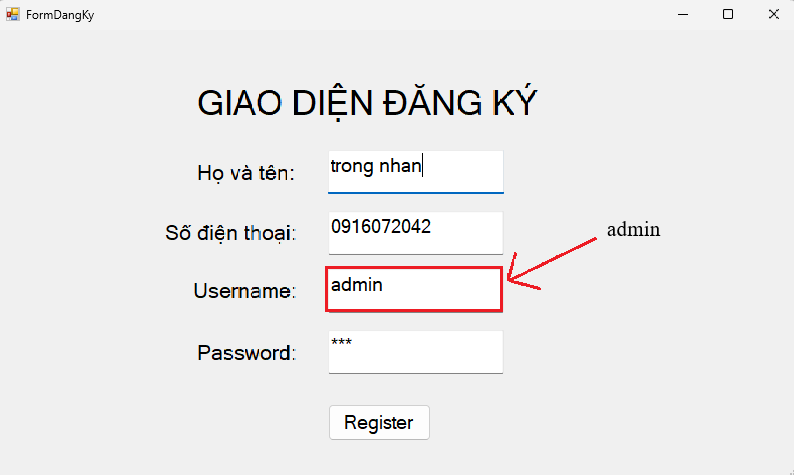
#### 2.2.2 CN2\_Register:

##### 2.2.2.1\_CheckRegisterDuplicateUsername\_TC1

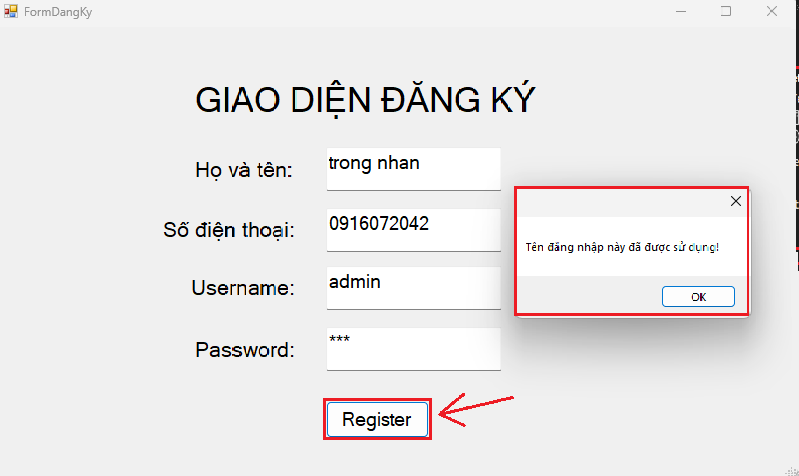
Mục đích: Kiểm tra đăng ký tài khoản trùng tên đăng nhập với 1 tài khoản có đã đăng ký

Tiền điều kiện: Đã có 1 tài khoản đăng ký với cái username đó được lưu trong account.json

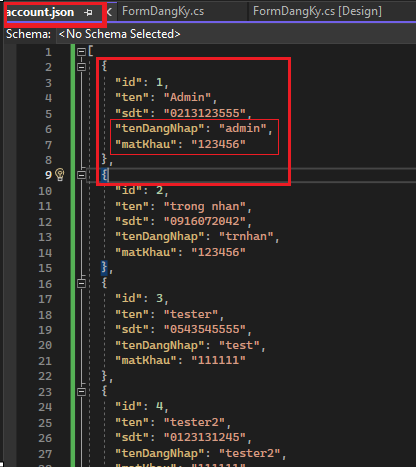
Kết quả mong đợi: Hệ thống sẽ hiển thị lỗi cho biết username đã tồn tại



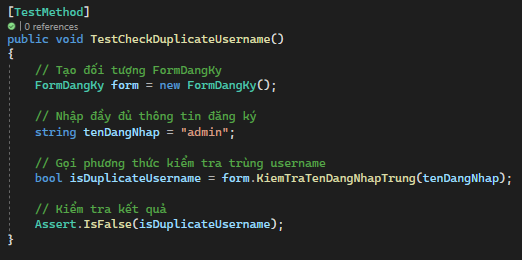
Bấm register sẽ hiển thị thông báo failed

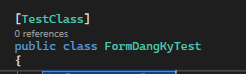


Username này đã có trong file json



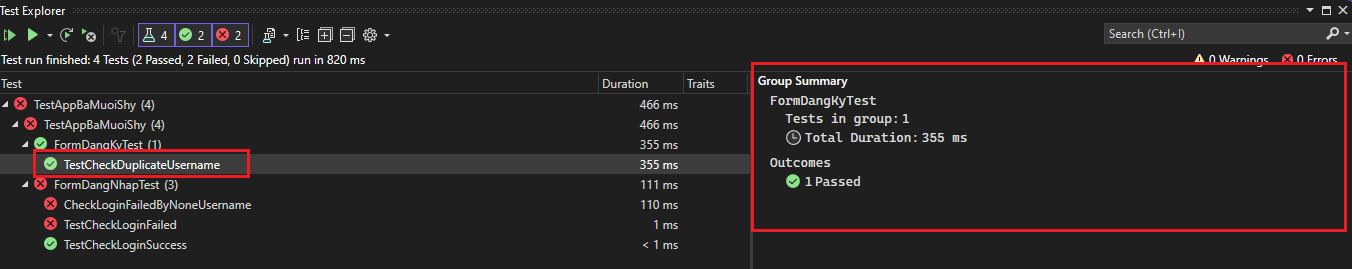
Viết testcase



Viết test case này bên trong TestClass khác  


Vì testcase sử dụng lệnh **Assert.IsFalse**, kết quả username hợp lệ là **false** nên nó trả kết quả **true**

Chạy testcase

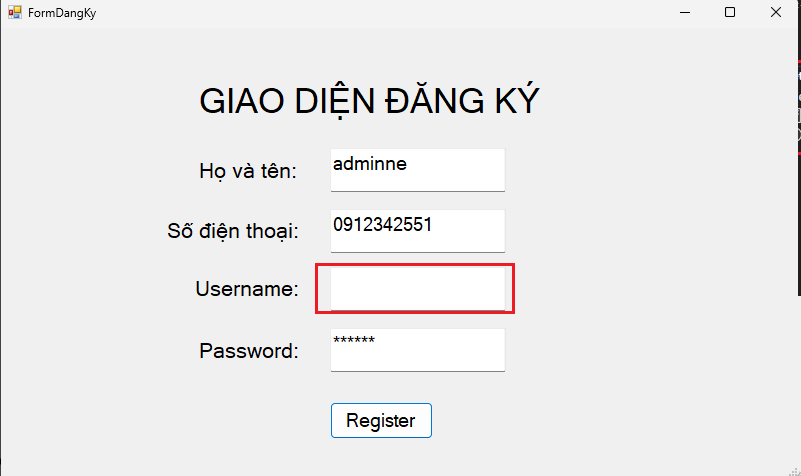


##### 2.2.2.2\_CheckRegisterFailedByMissingInfo\_TC2

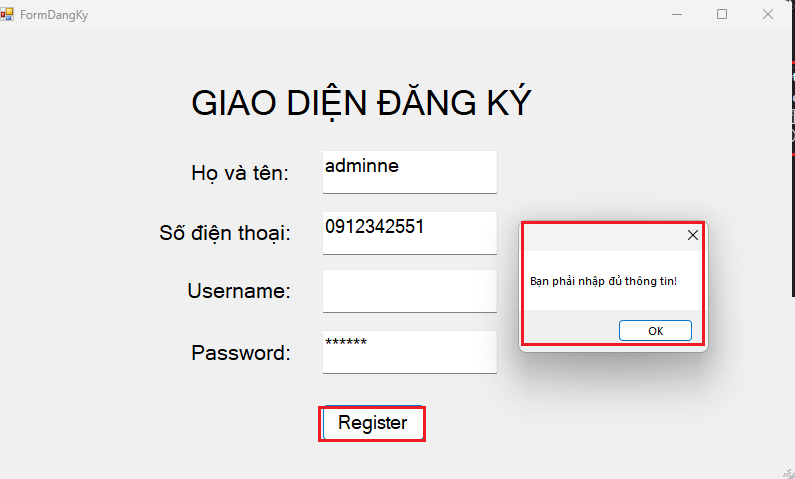
Mục đích: Kiểm tra đăng ký không được cho phép cho không nhập đủ thông tin

Tiền điều kiện: Thông tin không đủ để nhập vào nhưng vẫn đăng ký

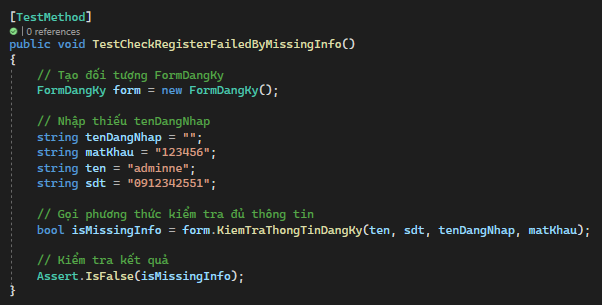
Kết quả mong đợi: Hệ thống sẽ hiển thị lỗi cho biết thiếu thông tin



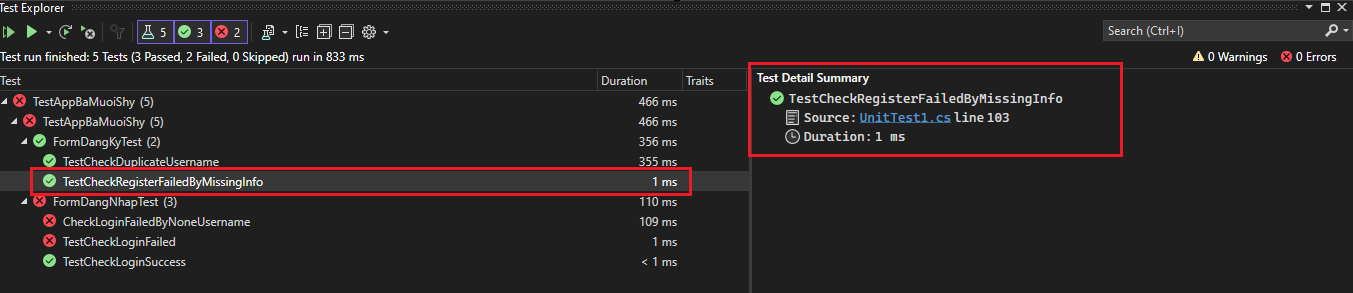
Ô username để rỗng nên khi bấm register sẽ hiện thông báo yêu cầu nhập đủ thông tin



Viết testcase



Vì testcase sử dụng lệnh **Assert.IsFalse**, kết quả username hợp lệ là **false** nên nó trả kết quả **true**

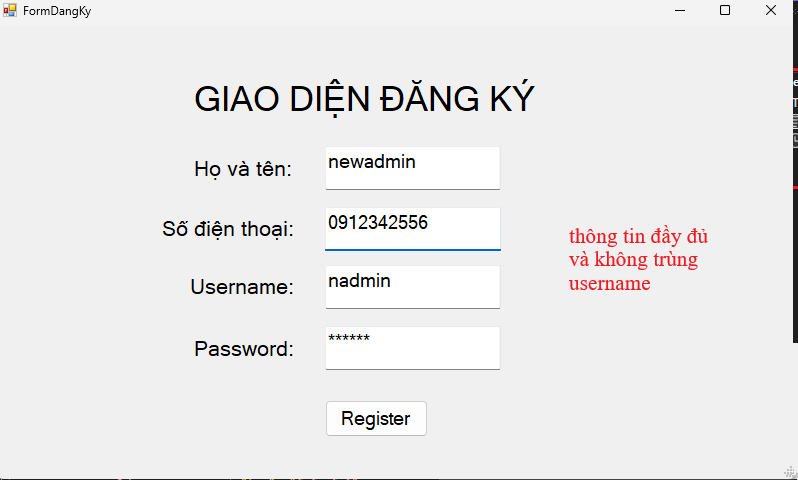


##### 2.2.2.3\_CheckRegisterSuccess\_TC3

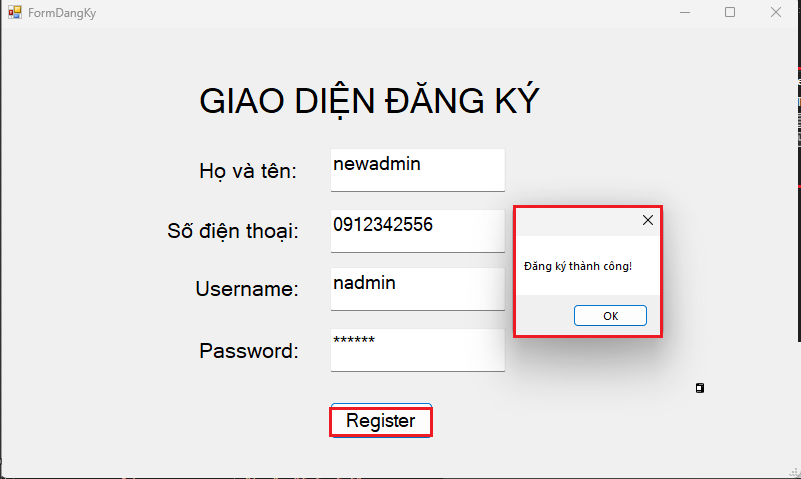
Mục đích: Kiểm tra đăng ký thành công

Tiền điều kiện: Nhập đầy đủ thông tin đăng ký hợp lệ

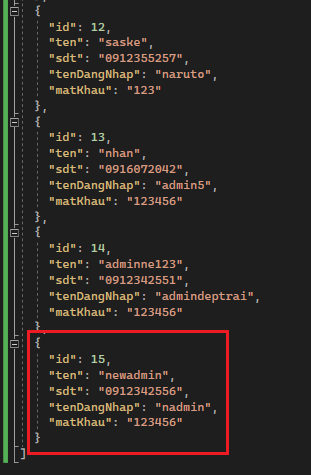
Kết quả mong đợi: Nếu nhập đầy đủ thông tin đăng ký hợp lệ, người dùng sẽ được đăng ký thành công



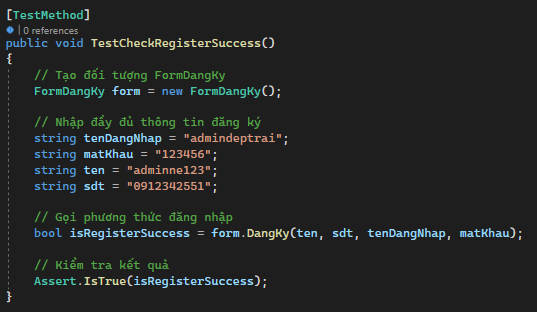
Click nút Register



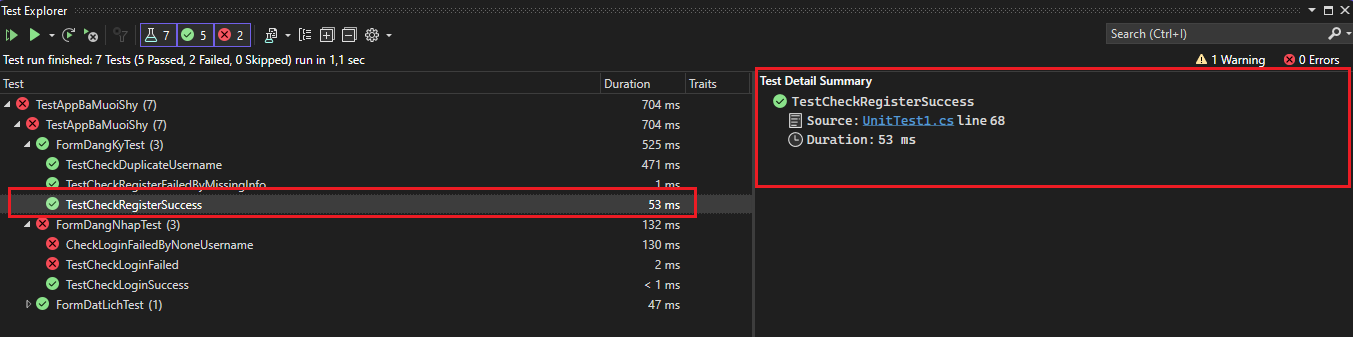
Account mới được lưu vào account.json



Viết testcase



Chạy testcase



#### 2.2.3 CN3\_Đặt Lịch:

##### 2.2.3.1\_CheckTestDatLichThanhCong\_TC1

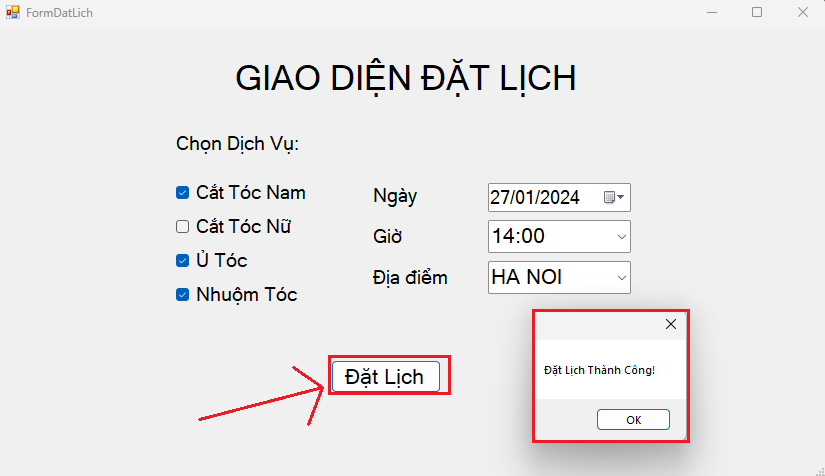
Mục đích: Kiểm tra đặt lịch thành công

Tiền điều kiện: Người dùng đã đăng nhập vào hệ thống thì mới đặt lịch được

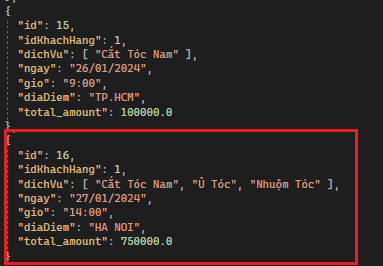
Kết quả mong đợi: Lịch được đặt thành công



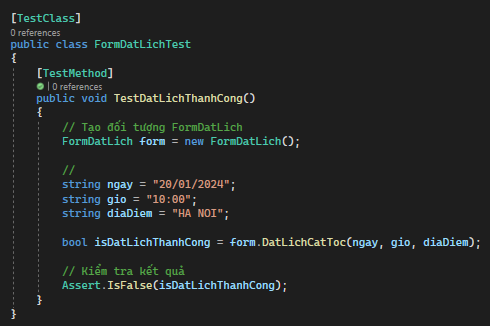
Bấm đặt lịch



File json lưu các lịch đặt



Viết test



Chạy testcase



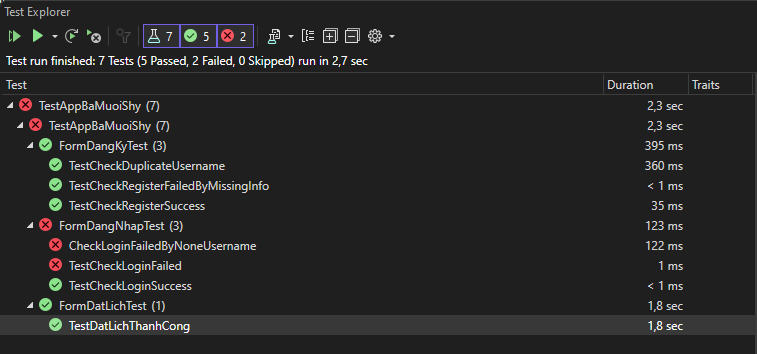
### 2.3 Kết luận

Ta có 7 testcase cho 3 chức năng

Chức năng Đăng Nhập – 3 TC – 1/3 Passed

Chức năng Đăng Ký – 3 TC – 3/3 Passed

Chức năng Đặt Lịch – 1 TC – 1/1 Passed

****

## CHƯƠNG 3. VIẾT CÁC TEST ĐỂ TEST CÁC APIs

### 3.1 Setup API

Sau khi có 2 file json lưu thông tin đặt lịch cắt tóc và danh sách người dùng thì ta sẽ viết APIs:

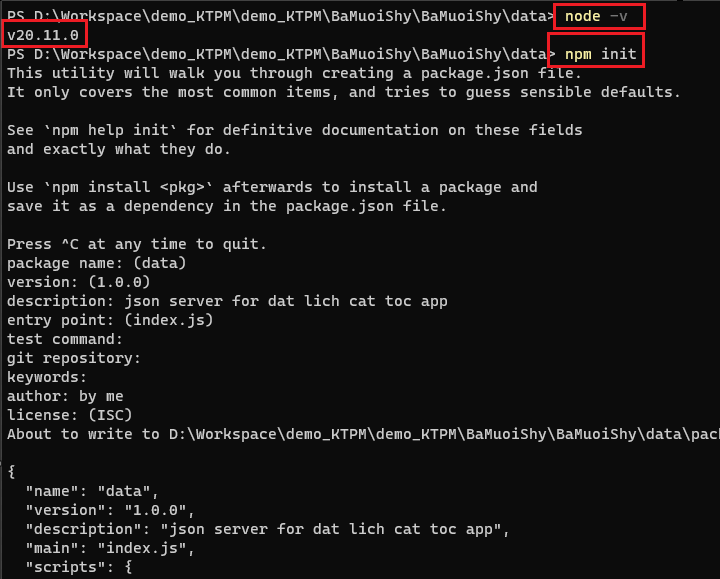
Tạo API GET danh sách account

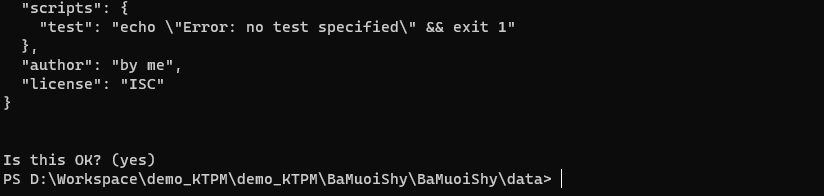
Tạo API GET id người dùng đã đặt lịch vào lúc mấy giờ, ngày nào và ở địa điểm nào

Tạo API DELETE lịch đặt

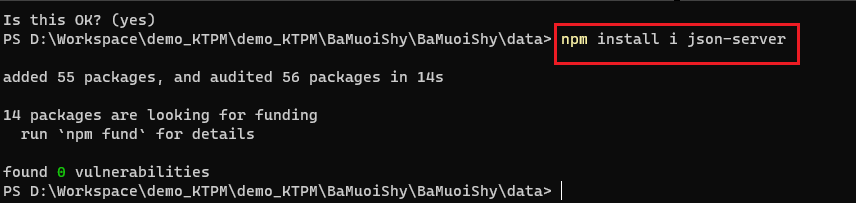
Tạo API POST lịch đặt mới

Bước 1: Cài nodejs và kiểm tra phiên bản nodejs để có thể tạo server json, dùng lệnh **npm -v** để kiểm tra phiên bản, khởi tạo node package manage ngay trong folder dự án của bản thân bằng lệnh **npm init**

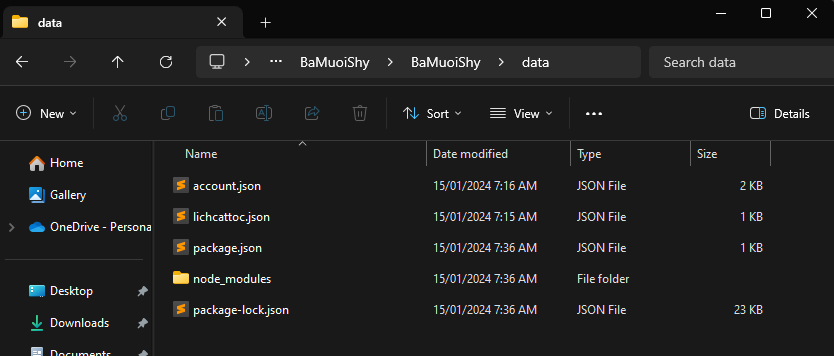




Sau đó gõ lệnh **npm install i json-server** để host 1 server json trong folder nơi khởi tạo npm

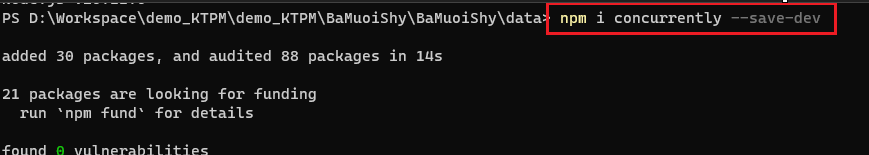


Kết quả có được trong folder

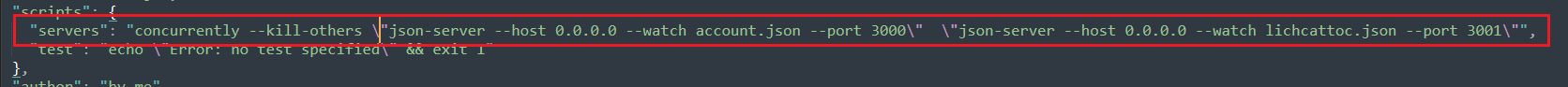


Bước 2: để host cả 2 file json thì ta làm như thế này

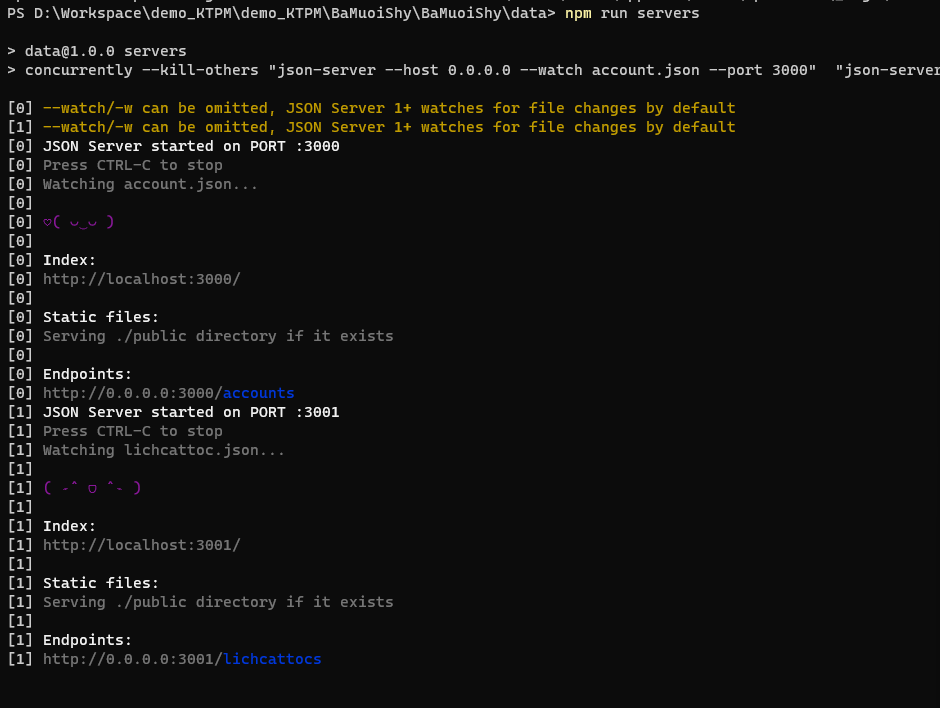
Ban đầu cài **concurrently** bằng lệnh **npm i concurrently –save-dev**



Sao đó mở file package.json thêm trường servers vào scripts như sau:

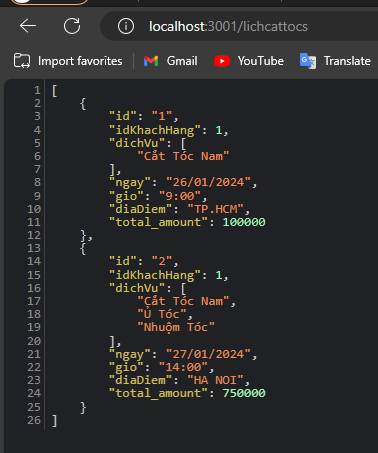


Cuối cùng là chạy server bằng lệnh **npm run servers**

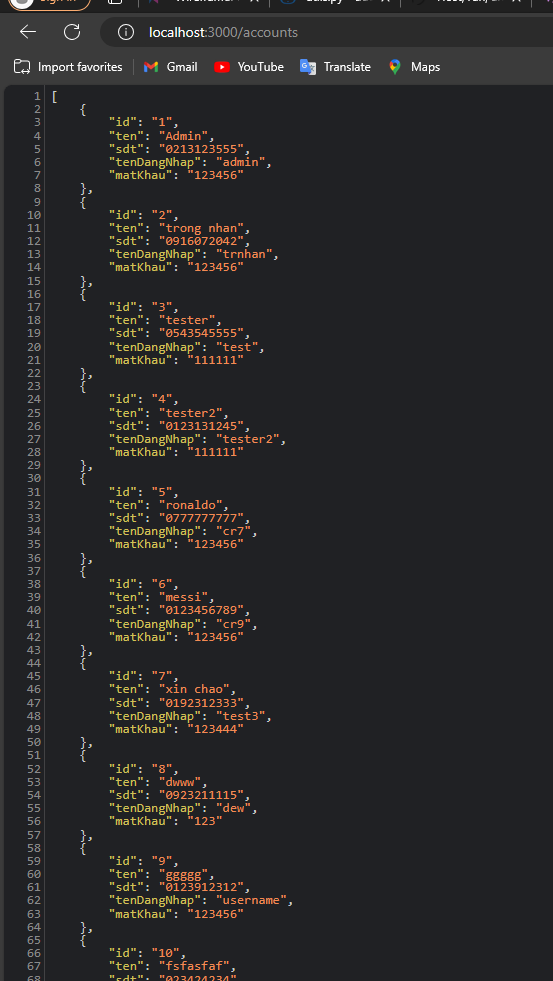


Cách này cơ bản là host 2 file json trên 2 port khác nhau là port 3000 và port 3001

Đây là port 3001 là port của file lichcattoc.json

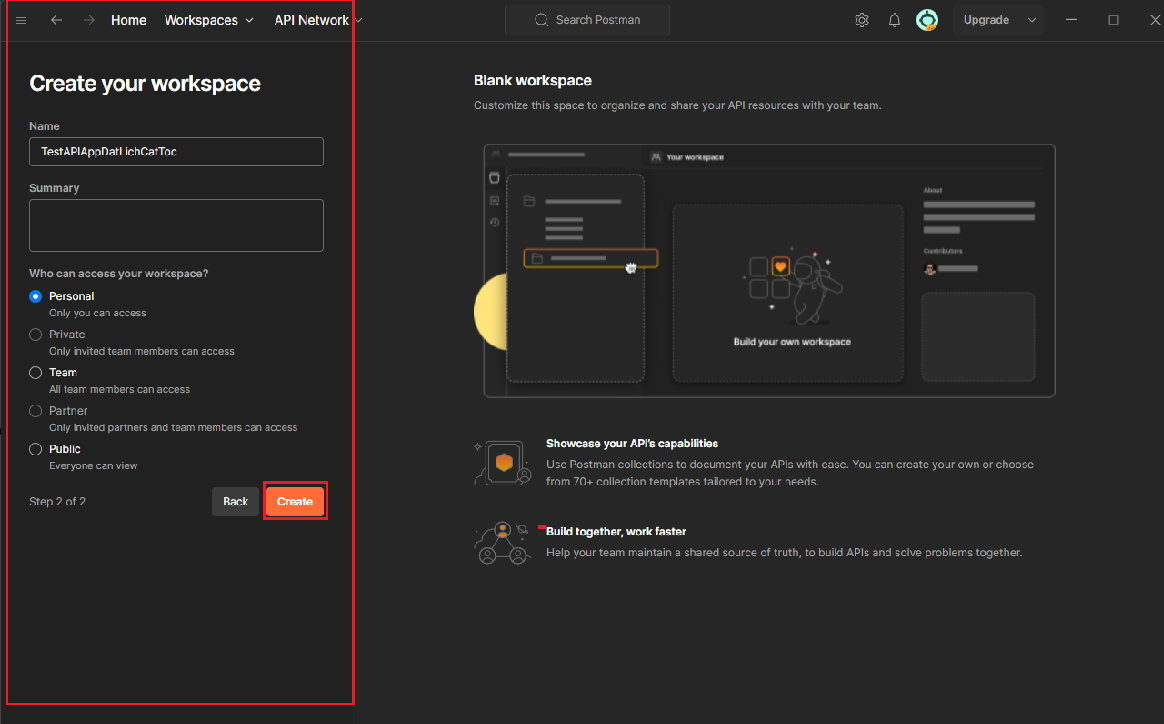


Đây là port 3000 là port của file account.json

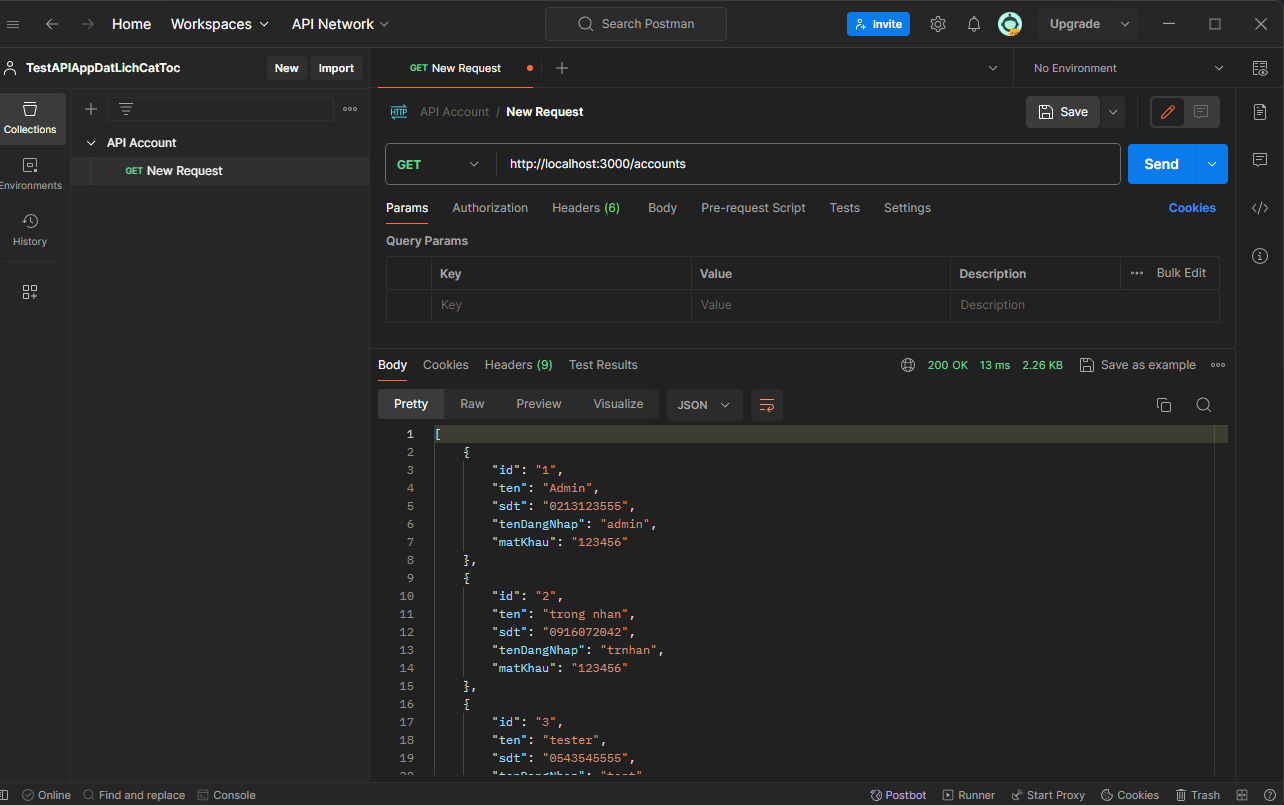


### 3.2 Test api

Lên post main tạo 1 workspace mới



Tạo collection dành cho các api account và chạy get api thử

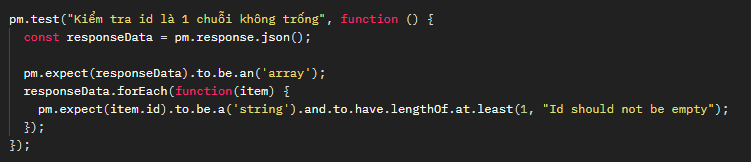


#### 3.2.1 Test API account phương thức GET

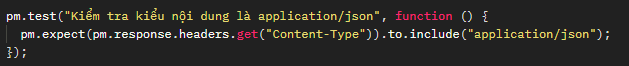
##### 3.2.1.1 TC-3.2.1.1: Kiểm tra trạng thái phải hồi



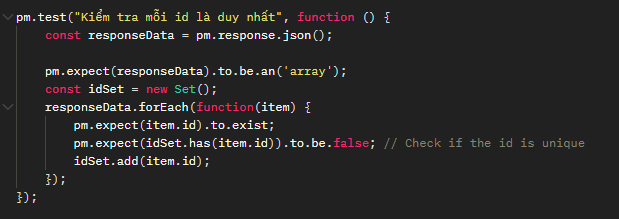
##### 3.2.1.2 TC-3.2.1.2: Kiểm tra id là 1 chuỗi không trống



##### 3.2.1.3 TC-3.2.1.3: Kiểm tra kiểu nội dung là application/json

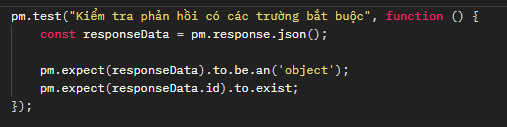


##### 3.2.1.4 TC-3.2.1.4: Kiểm tra mỗi id là duy nhất

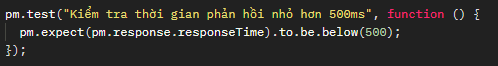


#### 3.2.2 Test API account phương thức POST

##### 3.2.2.1 TC-3.2.2.1: Kiểm tra phản hồi có các trường bắt buộc



##### 3.2.2.2 TC-3.2.2.2: Kiểm tra thời gian phản hồi nhỏ hơn 500ms



#### 3.2.3 Kết quả chạy collection API Account

