**LAB 10: Thực hành trích xuất dữ liệu từ phản hồi và chạy tự động các tệp câu lệnh**

**Giới thiệu tổng quan**

Bài lab này chúng ta sẽ tập trung vào 2 kỹ năng sau:

* Luyện tập kĩ năng bóc tách và kiểm thử tính đúng đắn của dữ liệu trả về từ API
* Thực hành chạy tự động Collection với Collection runner.

**Kiến thức liên quan:**

* Basic java script.
* Chuỗi và JSON.

**JSON** là chữ viết tắt của **Javascript Object Notation**, đây là một dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được. Nói một cách khác, JSON là định dạng giúp lưu trữ các thông tin có cấu trúc và nó chủ yếu được dùng để truyền tải dữ liệu giữa server và client.  
  
Định dạng JSON sử dụng các cặp **key – value** để dữ liệu sử dụng, các cặp key-value được bọc trong {} và được ngăn cách nhau bới dấu “,”

| {      "username" : "thehalfheart",      "email" : "thehalfehart@gmail.com",      "website" : "freetuts.net",      "title" : "Học lập trình với JSON"  } |
| --- |

Chúng ta có thể dễ dàng nhận ra rằng, Request Body trong method HTTP POST/PUT và response trả về của API đều có dạng JSON object.  
VD: Để lấy thông tin các trường từ response của API, chúng ta có thể sử dụng cú pháp sau: response.username, response.email….

**I/ Yêu cầu thực hành**

1. Thực hành viết script để kiểm tra tính đúng đắn dữ liệu trả về.
2. Lưu lại các API đã khởi tạo thành một Collection.
3. Sử dụng Collection runner để chạy tập API.

**II/ Ví dụ mẫu**

**Đề bài.**

Cho danh sách API sau:

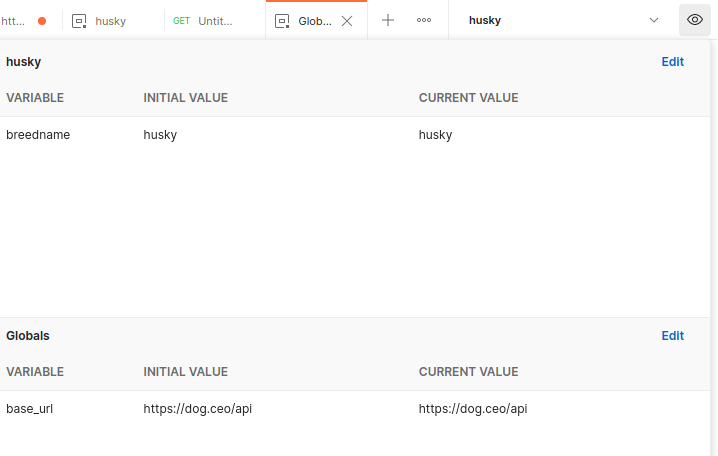
| **API** | **Param/response** | **Requirement** |
| --- | --- | --- |
| GET:  <https://dog.ceo/api/breeds/list/all>  Mô tả: API trả về danh sách các giống chó | response  {  message :  (danh sách giống chó và các dòng của giống)  status :   (success | failure )  } | - Nếu thành công, API cần trả về HTTP 200 OK   response sẽ chứa thông tin status = “success” và có trường message  - Nếu Không thành công, API có thể trả về response code khác 200 Ok hoặc status = “failure”. |
| GET: [https://dog.ceo/api/breed/:breedname](https://dog.ceo/api/breeds/image/random)/images/random  Mô tả: API trả về ngẫu nhiên hình ảnh một chú chó thuộc giống {breedname} được cung cấp | Path params: breedname: required. (tên giống chó)   response  {  message :  (đường link ảnh chú chó)  status :   (success | failure )  } | - Nếu thành công, API cần trả về HTTP 200 OK   response sẽ chứa thông tin status = “success” và có trường message  - Nếu Không thành công, API có thể trả về response code khác 200 Ok hoặc status = “failure”.  - Nếu breedname không tồn tại, APi trả về 404 not found nội dung response cần có thông tin sau : message : "Breed not found (master breed does not exist)" |

Yêu cầu:

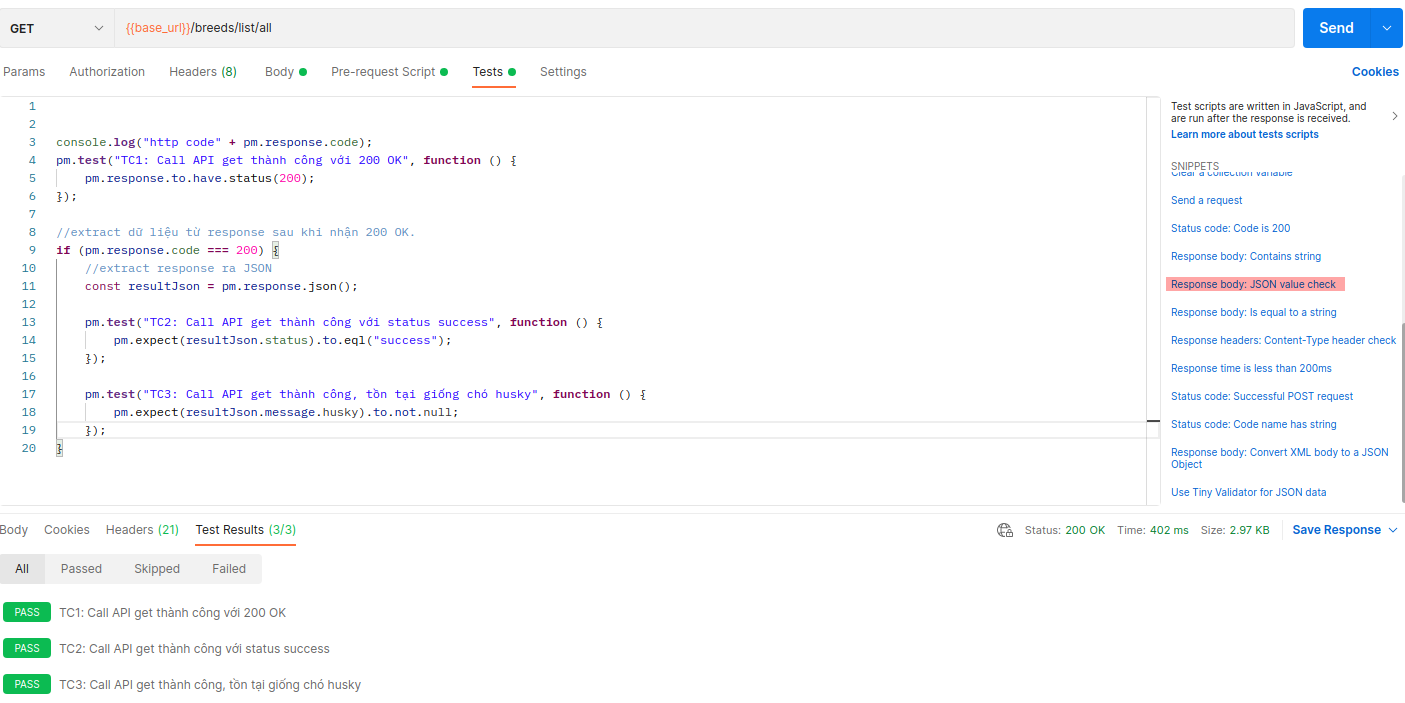
**-** Kiểm tra dữ liệu trả về có giống chó “husky” hay không.  
 - Nếu có giống chó “husky”, thực hiện kiểm thử API lấy ra ngẫu nhiên hình ảnh một chú chó husky.

**Thực hành.  
  
1. Thiết lập các biến môi trường.**

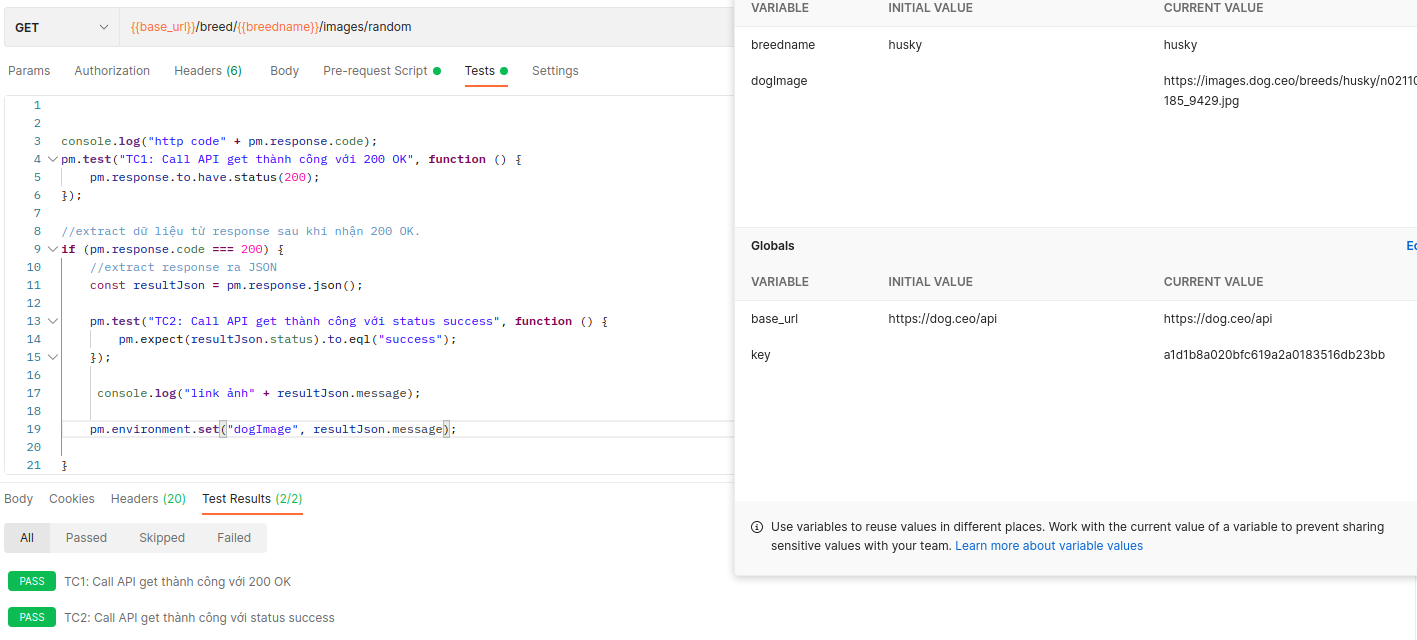
Global: **base\_url** : <https://dog.ceo/api>  
 Enviroment : **breedname** : husky



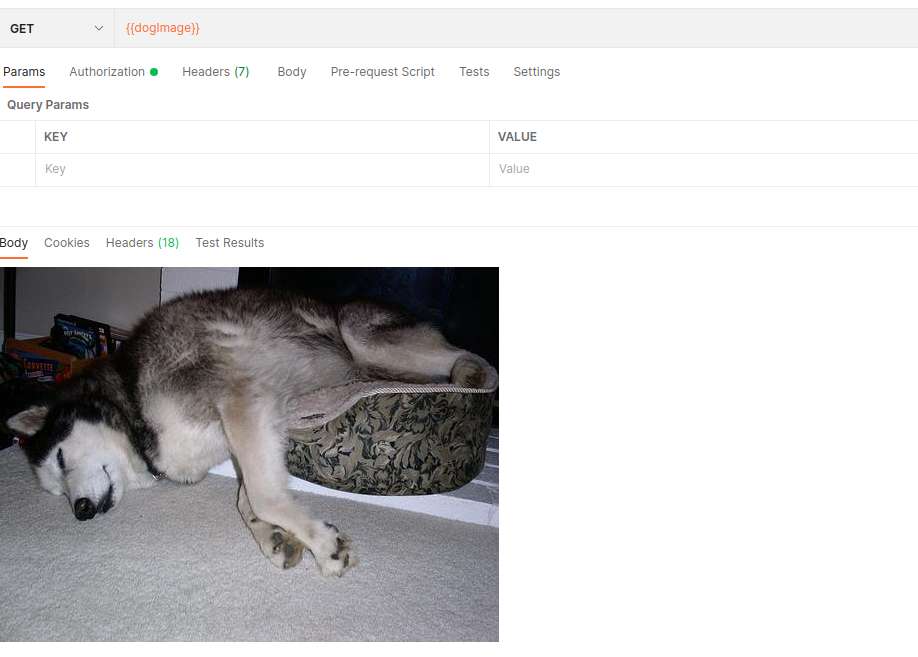
**2.Viết các script kiểm thử api lấy danh sách giống chó**

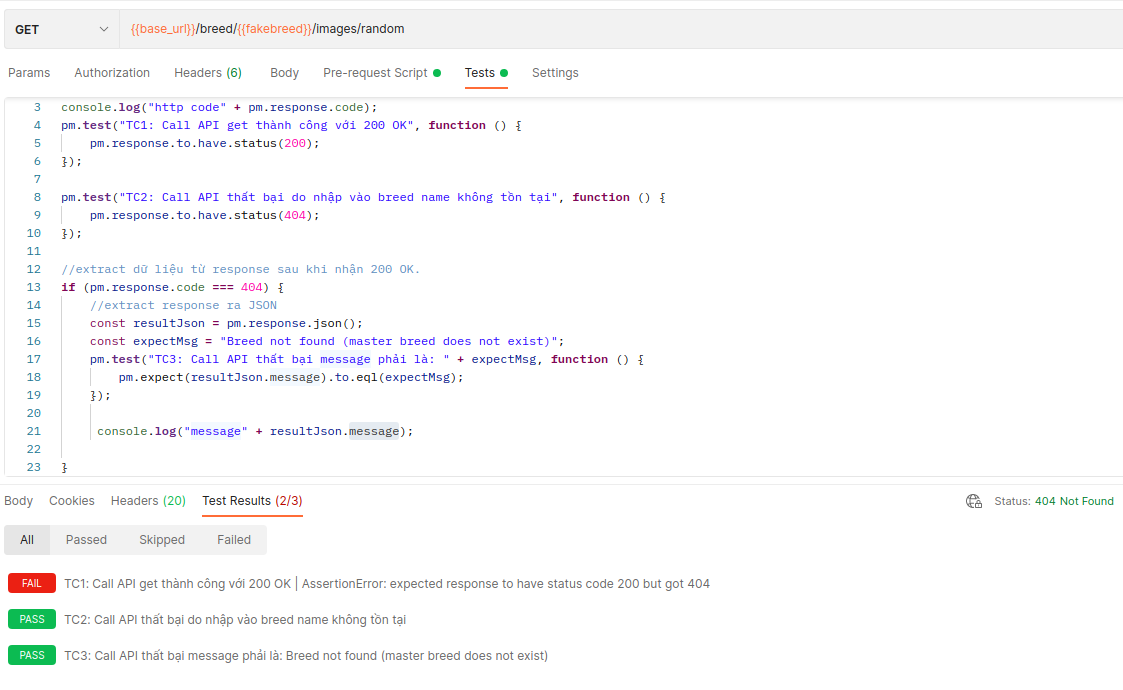


**3. API lấy random một bức ảnh chó husky.** Sau khi nhận response, lưu đường dẫn ảnh vào biến môi trường **dogImage**

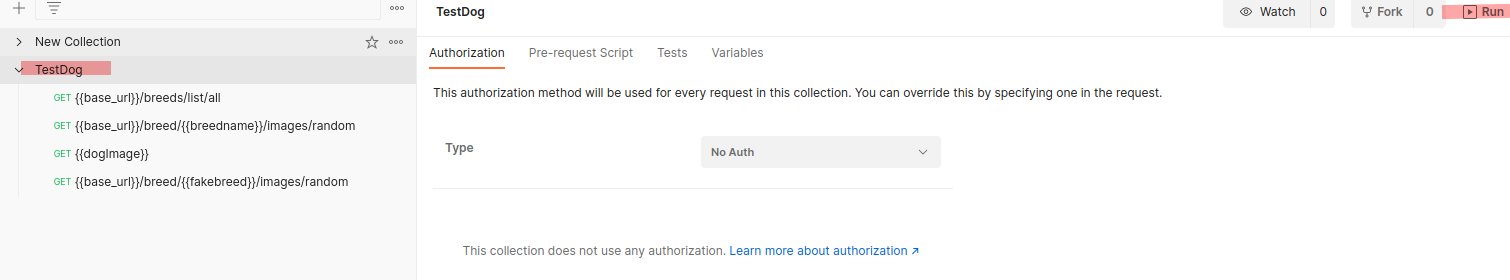


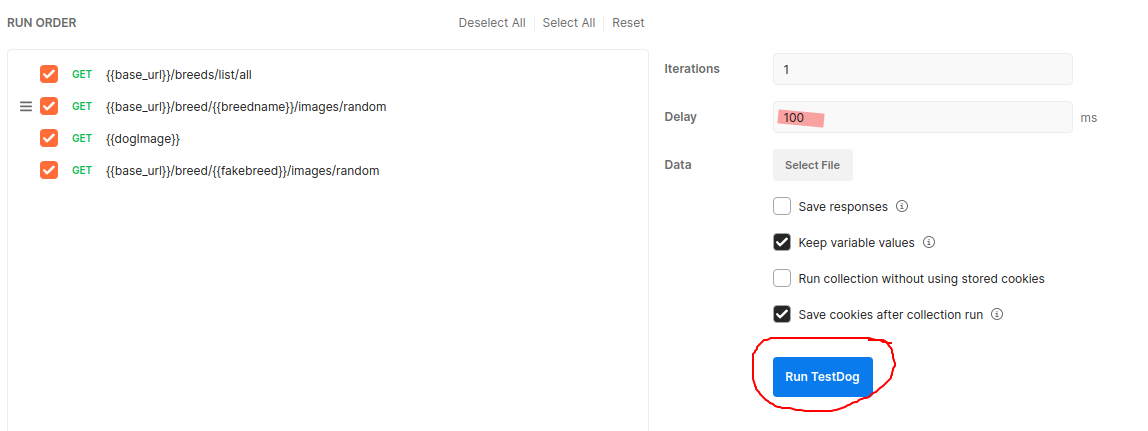
**4. Call API GET dữ liệu ảnh với link ảnh trả về ở mục #3**

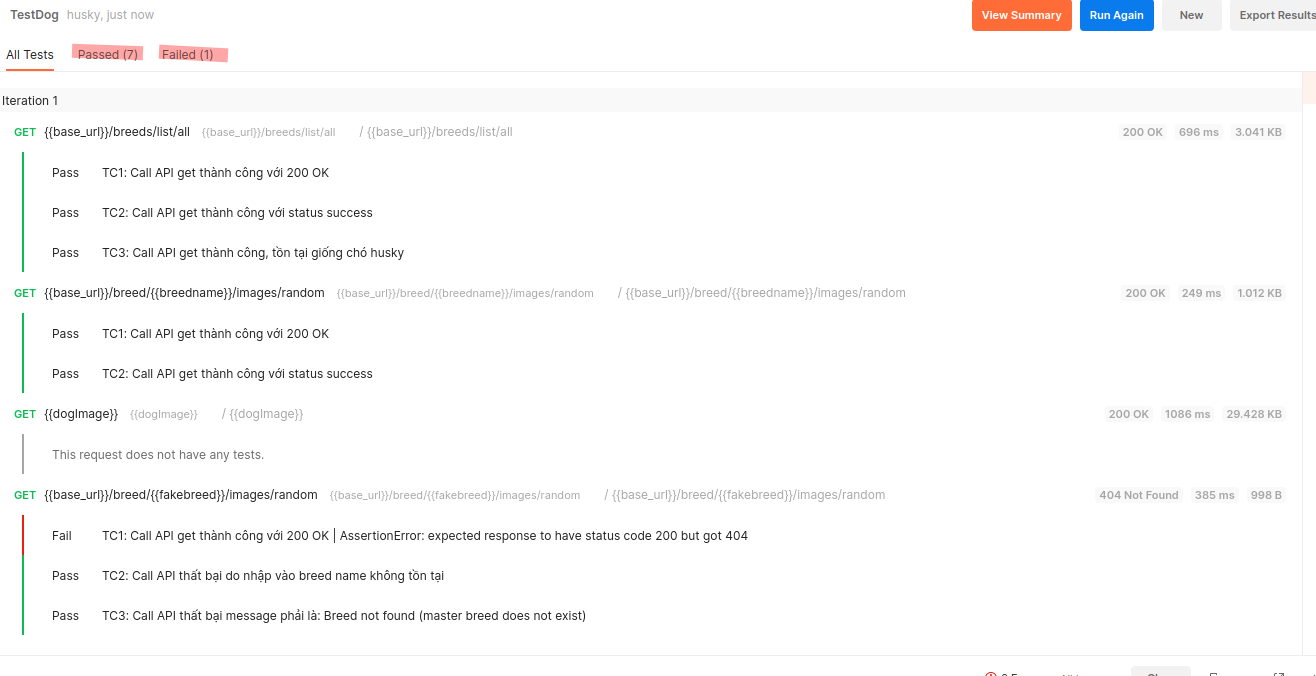
**5. Request check trường hợp nhập vào breedname không tồn tại**

Tạo một biến môi trườngbreed fake có tên **fakebreed : abc**  
  


**6. Sử dụng Collection runner để chạy tự động cả collection.**  
  
Sau khi viết testcase cho các API, chúng ta sẽ tạo ra một collection như dưới đây, sau đó chọn Run với Runner.



Chọn số lần thực thi và thời gian delay giữa các request.  


**Kết quả:**  


**III/ Học viên thực hành.**

**Đề bài.**  
**Danh sách API.**

| **API** | **Param/response** | **Requirement** |
| --- | --- | --- |
| **GET:**  https://api.thecatapi.com/v1/breeds  Mô tả: API trả về danh sách các giống mèo cũng như mô tả, tính cách của từng giống. | response  [{  id : id (4 kí tự)  name: tên giống.  temperament : tính cách  origin: Nguồn gốc  description : mô tả về giống mèo  wikipedia\_url : link wiki  }  ] | - Nếu thành công, API cần trả về HTTP 200 OK   response sẽ là một mảng các giống mèo, mỗi một giống cần có đầy đủ thông tin các trường id, name, temperament, origin, description, wikipedia\_url  - Nếu Không thành công, API có thể trả về response code khác 200 Ok . |
| **GET:** https://api.thecatapi.com/v1/images/search?breed\_ids={breed-id}  Mô tả: API trả về ngẫu nhiên hình ảnh một chú mèo thuộc giống {breed-id} được cung cấp | **request params:** breed-id: required. (id của giống mèo, lấy từ kết quả của API bên trên)   **response**  {  breeds :   (mảng các breed)  id : id ảnh  url : đường link download ảnh  width : độ chiều rộng của ảnh  height: chiều dài ảnh  } | - Nếu thành công, API cần trả về HTTP 200 OK  - Nếu Không thành công, API có thể trả về response code khác 200 .  - Nếu breed-id không tồn tại, APi trả về 404 not found nội dung response cần có thông tin sau : message : "Breed not found (master breed does not exist)" |
| **POST**  <https://api.thecatapi.com/v1/votes>  Mô tả: API tạo một user vote User like/dislike bức ảnh | Body:  {  "image\_id": id của image lấy từ API #2,  "value": 1|0 tương ứng với Like và dislike  "sub\_id": "Id của người vote, chúng ta sẽ tạo 1 id bất kì"  }  Response  {  "message": "success| failure",  "id": id của vote  } | - Nếu thiếu image\_id, API cần trả về 400 Bad request.  -Nếu nhập vào một image\_id không hợp lệ, cần trả về 404 Not found.  -Nếu thiếu trường value. API trả về 400 Bad request và message sẽ phải là  "\"value\" is required"  -Nếu trường sub\_id không có hoặc để trống, API trả về 400 Bad request và message sẽ phải là  "\"sub\_id\" is not allowed to be empty"  -Nếu thành công thì trả về 201 Create và message là “success” cùng với id của vote. |
| **GET**  https://api.thecatapi.com/v1/votes?sub\_id={userId}  Mô tả: Lấy ra danh sách votes theo user | **API yêu cầu AUTHEN.** cần add **x-api-key** vào header. Tạm thời chúng ta sẽ sử dụng key sau để test: **DEMO-API-KEY**  **request param order:** optional(ASC|DESC sắp xếp tăng, giảm)  **limit:** optional giới hạn số lượng vote lấy về)  **response** : mảng  [  {  "id": id vote,  "image\_id": id ảnh,  "sub\_id": id user,  "created\_at": "2018-10-24T08:36:13.000Z",  "value": 1,  "country\_code": **null**  },  ….. ] | -Nếu thành công thì trả về 200 Ok. và danh sách votes của userId đó.  Việc tạo vote phải cần có userId, do đó nếu :  -Nếu để userId trống, API cần trả về 400 Bad request với message là:  “User is not allowed to be empty"  - Nếu vào userId không tồn tại thì API trả về 404 Not Found |

* Kiểm chứng thông tin 2 giống mèo có tên (name) là “American Bobtail”, “Arabian Mau” có trong cơ sở dữ liệu hay không
* Nếu có, với mỗi giống mèo, lấy ra 2 bức ảnh
* Tạo 1 userId bất kì.
* Tạo 2 vote, 1 like, 1 dislike cho mỗi ảnh.
* Lấy ra danh sách vote của user

**Yêu cầu:**

1. Viết script test cho các API và các requirement để kiểm tra kết quả api có đúng với các yêu cầu trong mô tả.
2. Tạo một collection chứa các request và thực thi collection tự động bằng Collection Runner.

**Gợi ý:**  
1. Đối với yêu cầu kiểm tra kết quả api có đúng với requirement, các bạn có thể tạo testcase hoặc checklist

2. Tạo các biến global, environment để lưu thông tin như : url, tên giống cần kiểm định, authen key, userId.  
 Các id của giống mèo và id ảnh thì set động sau khi có kết quả API trả về.

3. Viết Script kiểm thử API lấy về các giống mèo, lưu lại id giống nếu cần

4. Viết Script kiểm thử API lấy về ảnh mèo với giống tương ứng, lưu lại id ảnh nếu cần

5. Tạo vote cho mỗi ảnh, lấy danh sách vote theo user.

6. Tạo và thực thi collection

*Một số chú ý:*

Response có thể trả về 1 mảng dữ liệu:  
Ví dụ  
  
[

{

"id": "abys",

"name": "Abyssinian",

},

{

"id": "sfsfd",

"name": "hdfdfdgd",

}

]

=> để lấy dữ liệu ta có thể làm như sau:

| //lấy ra response dạng JSON  const result **=** pm.response.json();  //Lấy ra độ dài phần tử của mảng const length **=** result.length; // với vd trên thì =2  //lấy ra 2 phần tử đầu của mảng  **for** (var i **=** 0; i**<** 2; i**++**) {  console.**log**(result[i]); // in log ra cửa sổ console  // kiểm tra các trường name của object có giá trị hay không  **if** (result[i].name **!=** **null**) {  console.**log**(result[i].name); // in log ra cửa sổ console  }  }  // kiểm tra phần tử đầu tiên có đúng là Abyssinian hay không  pm.**test**("TC Name must be Abyssinian", **function** () {  pm.expect(result[i].name).to.eql(“Abyssinian”);  });  Console |
| --- |