

**Nama** : Muhammad Azhar Rasyad

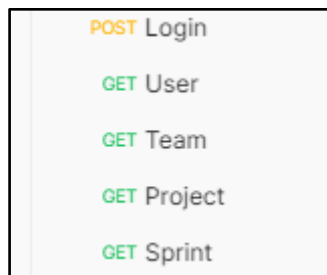
**NIM** : 0110217029

---

## **Penerapan *Test Automation* untuk Web**

### **Pendahuluan**

Dalam makalah ini akan membahas mengenai penerapan tes secara otomatis pada platform web. Web yang digunakan untuk pengujian adalah Link-Match STT-NF, web tersebut merupakan penelitian tugas akhir oleh peneliti. Pengujian akan dilakukan terhadap *web service* dengan REST API sehingga salah satu *tools* yang peneliti gunakan yaitu Postman. Terdapat beberapa pengujian yang akan dilakukan diantaranya terlihat pada gambar berikut.



Gambar 1. Pengujian REST API

Terlihat pada gambar di atas, ada 5 pengujian yang akan dilakukan diantaranya 1 untuk *auth* dan 4 lainnya untuk mengambil data yang telah tersedia. Namun sebelum menerapkan tes secara otomatis, perlu peneliti lakukan secara manual terlebih dahulu untuk memastikan bahwa tes yang dilakukan berhasil atau tidak. *Web service* yang digunakan memiliki URL berikut:

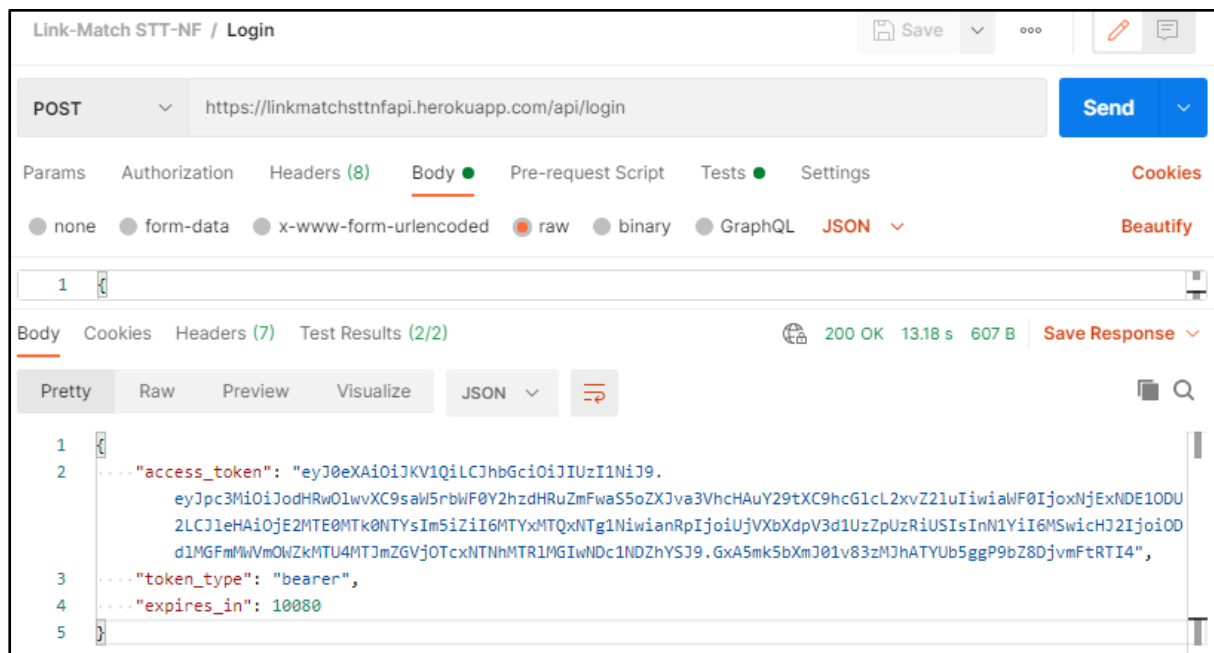
- <https://linkmatchsttnfapi.herokuapp.com/api/login> [POST]
- <https://linkmatchsttnfapi.herokuapp.com/api/user> [GET]
- <https://linkmatchsttnfapi.herokuapp.com/api/team> [GET]
- <https://linkmatchsttnfapi.herokuapp.com/api/project> [GET]
- <https://linkmatchsttnfapi.herokuapp.com/api/sprint> [GET]

Adapun web yang digunakan memiliki URL berikut

- <http://linkmatchsttnfweb.herokuapp.com/>

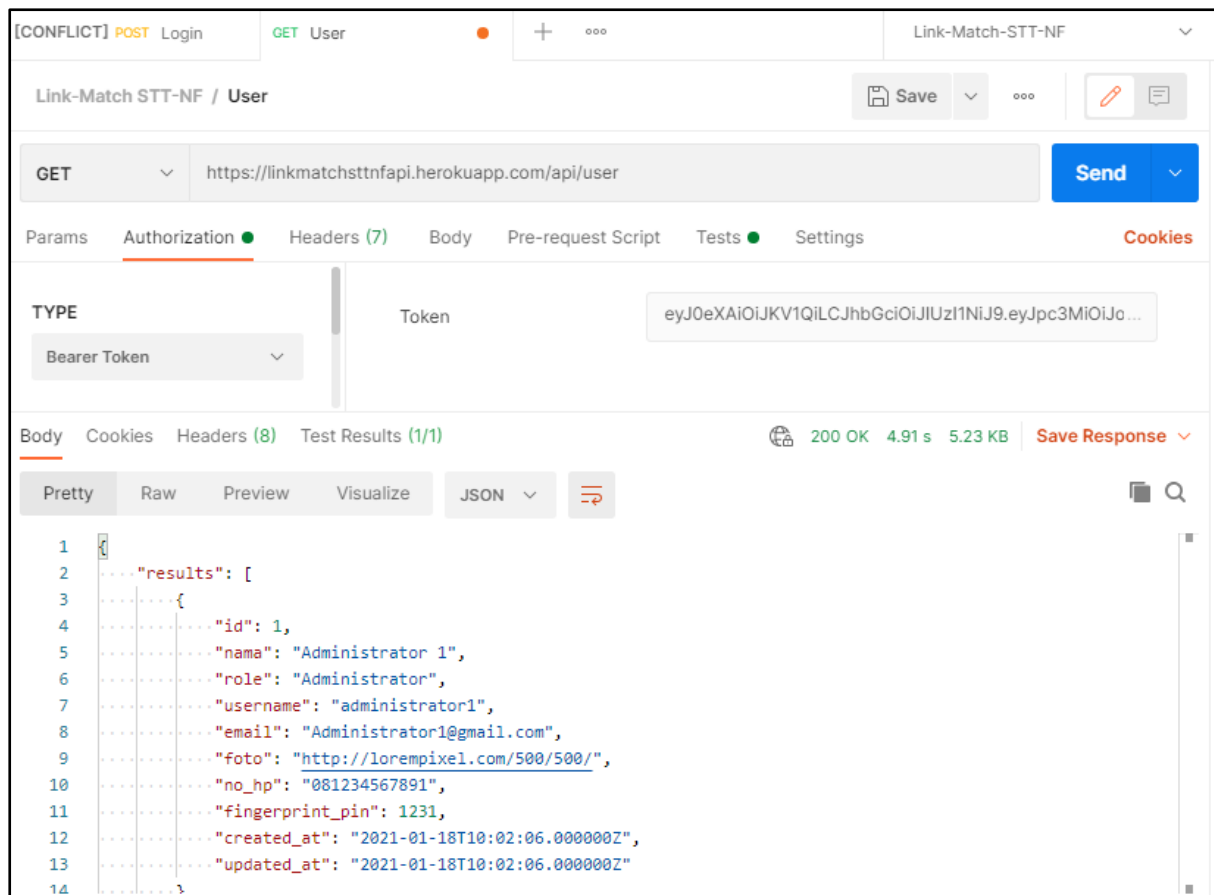
## Pembahasan

Langkah awal yang dilakukan yaitu menguji bagian *auth* karena jika bagian ini tidak berhasil maka *web service* lainnya tidak akan bisa diakses. Pengujian pada *auth* atau *login* memiliki beberapa tahapan yaitu menggunakan *method* POST, mengisi URL <https://linkmatchsttnfapi.herokuapp.com/api/login>, mengisi *username* serta *password* dengan tepat, dan terakhir yaitu *send* seperti pada gambar berikut:



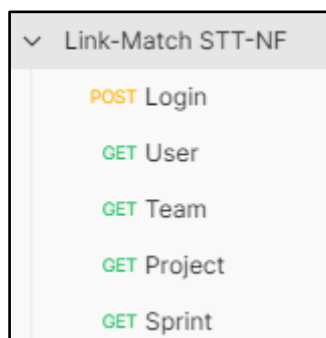
Gambar 2. Pengujian *Login*

Terlihat dari gambar di atas bahwa pengujian *login* berhasil dilakukan dengan ditandai status 200 dan mendapatkan token. Token tersebut nantinya akan digunakan pada bagian ambil data sehingga hanya pihak yang memiliki token untuk dapat mengakses data-data didalamnya. Pengujian selanjutnya akan dilakukan pada data *user* dengan melalui beberapa tahapan yaitu menggunakan *method* GET, mengisi URL <https://linkmatchsttnfapi.herokuapp.com/api/user>, memasukkan token pada menu *Authorization* dengan memilih Bearer Token, dan terakhir yaitu *send* seperti gambar berikut:



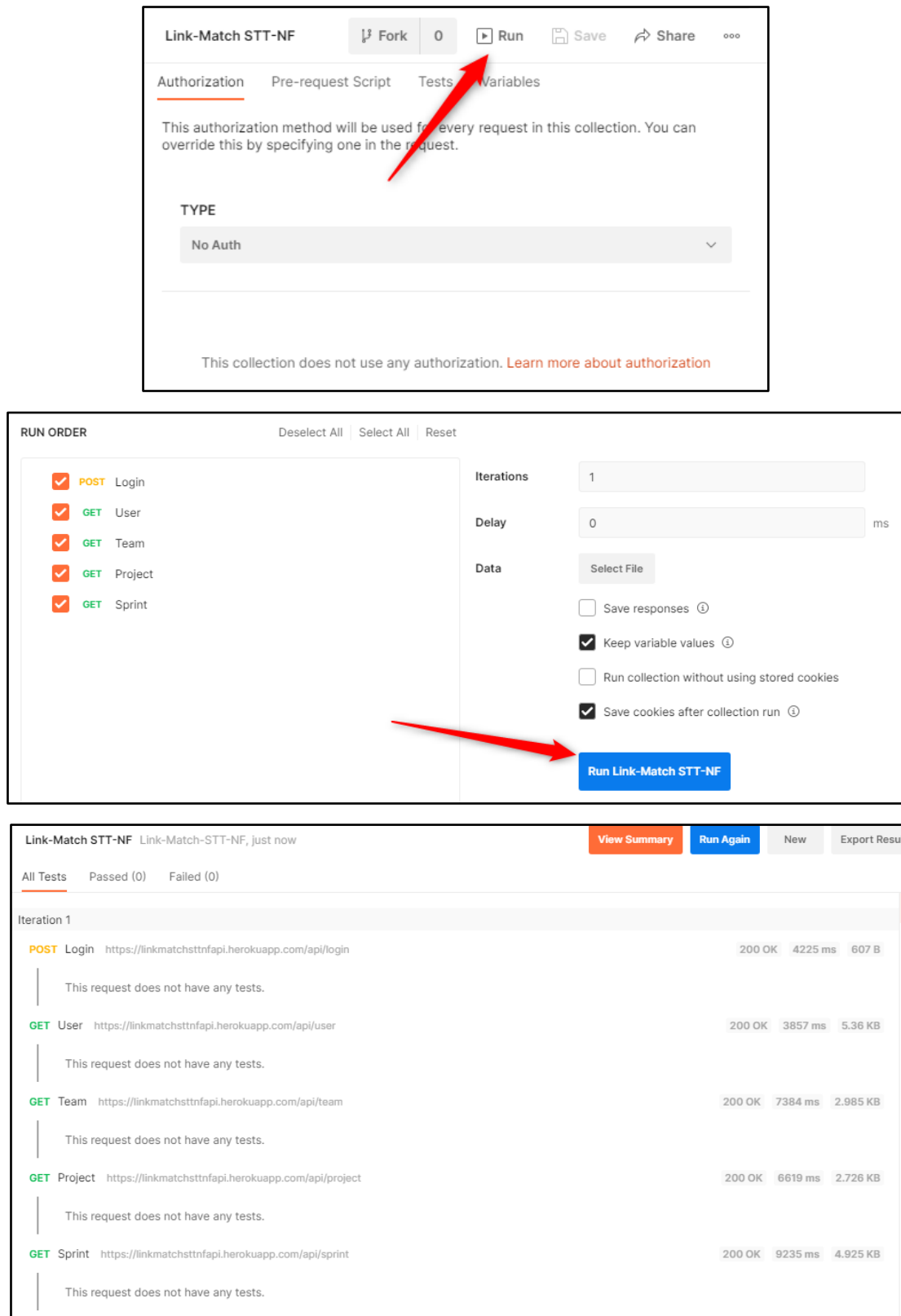
Gambar 3. Pengujian Data *User*

Terlihat lagi pada gambar di atas bahwa data yang kita *request* telah berhasil diakses dengan ditandai status 200 dan menampilkan data-data *user*. Pengujian pada data *team*, *project*, dan *sprint* sama dengan data *user*, selanjutnya ketika sudah berhasil melakukan tes secara manual hanya tinggal mengubahnya menjadi otomatis. Cara agar membuat Postman dapat menjalankan langkah-langkah sebelumnya yaitu dengan memasukkannya ke dalam *collection*. *Collection* ini nantinya yang akan menguji *request* berdasarkan isinya, berikut gambaran dari *collection* yang akan digunakan.



Gambar 4. *Collection* Link-Match STT-NF

Setelah dimasukkan ke dalam *collection* maka cukup menjalankan fungsi *Run* yang telah disediakan oleh Postman seperti gambar berikut.



Gambar 5. *Run Collection*

Terlihat bahwa pengujian *web service* pada setiap URL telah berhasil dilakukan dengan ditandai status 200, akan tetapi masih belum terlihat apakah *request* yang dilakukan berhasil atau tidak. Cara agar mengetahui hal tersebut yaitu dengan menambahkan *script* pada menu *Test* di masing-masing URL seperti pada gambar berikut.

The image contains two screenshots of the Postman interface showing test results for two different API endpoints.

**Top Screenshot: Link-Match STT-NF / Login**

- Method:** POST
- URL:** `https://linkmatchsttnfapi.herokuapp.com/api/login`
- Tests Tab:** Contains two test scripts:

```
1 pm.test('Login Berhasil', function(){
2   ...pm.response.to.have.status(200);
3 });
4
5 pm.test('Ambil Akses Token', function(){
6   ...const {access_token} = pm.response.json();
7   ...pm.environment.set('access_token', access_token);
8   ...return 'access_token';
9 });
```
- Test Results (2/2):** Shows two passed tests:
  - PASS Login Berhasil**
  - PASS Ambil Akses Token**
- Status:** 200 OK

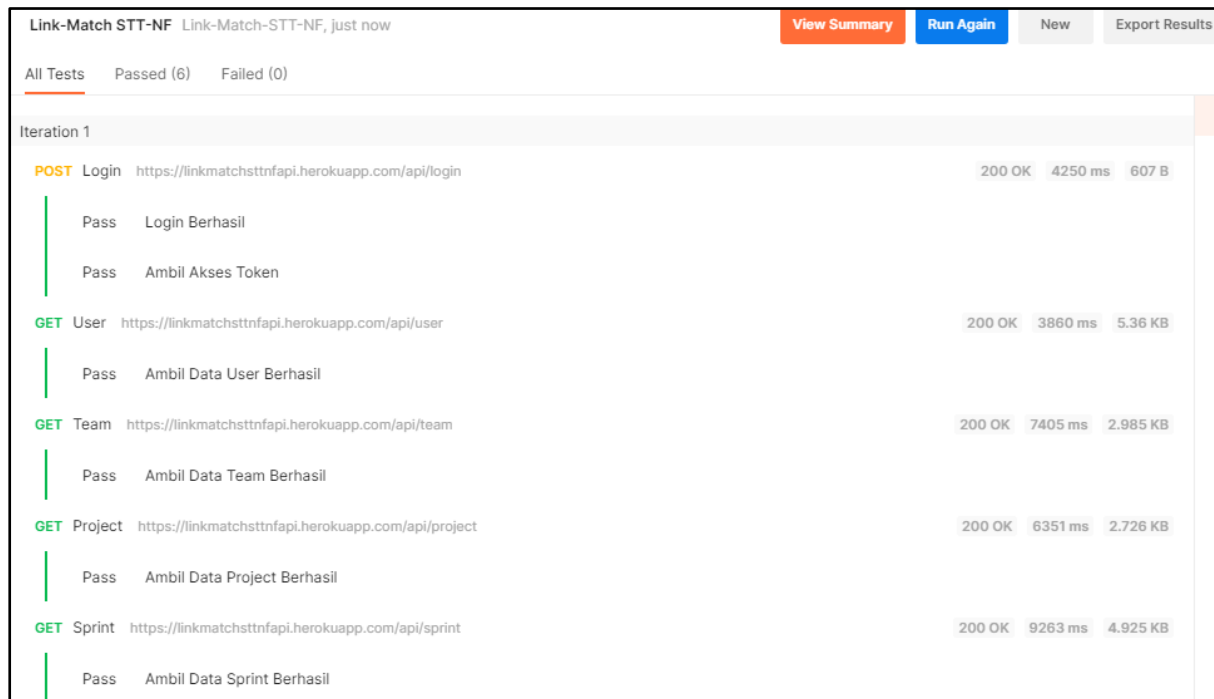
**Bottom Screenshot: Link-Match STT-NF / User**

- Method:** GET
- URL:** `https://linkmatchsttnfapi.herokuapp.com/api/user`
- Tests Tab:** Contains one test script:

```
1 pm.test("Ambil Data User Berhasil", function () {
2   ...pm.response.to.have.status(200);
3 });
```
- Test Results (1/1):** Shows one passed test:
  - PASS Ambil Data User Berhasil**
- Status:** 200 OK

Gambar 6. *Script Tests*

Sehingga hasil yang sebelumnya tidak diketahui apakah *request* yang diminta berhasil atau tidak, dapat terlihat seperti gambar berikut.

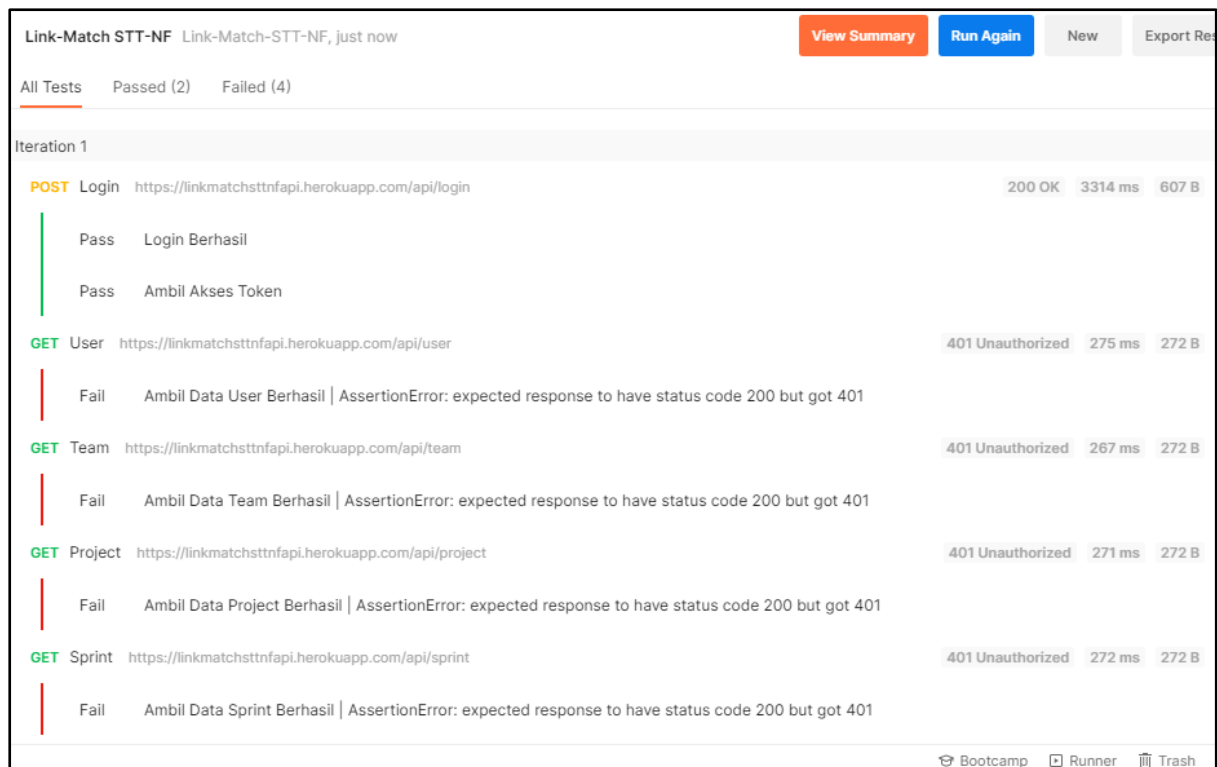


The screenshot displays a test automation interface for a project named 'Link-Match STT-NF'. The interface shows a summary of test results for 'Iteration 1', indicating that all tests passed. The tests include a POST login request and several GET requests for user, team, project, and sprint data. Each test step is marked as 'Pass' with a green vertical bar. The interface also provides details such as the HTTP status (200 OK), response time, and response size for each request.

Test Step	Method	URL	Status	Response Time	Response Size
POST Login	POST	https://linkmatchsttnfapi.herokuapp.com/api/login	200 OK	4250 ms	607 B
Pass Login Berhasil	Pass				
Pass Ambil Akses Token	Pass				
GET User	GET	https://linkmatchsttnfapi.herokuapp.com/api/user	200 OK	3860 ms	5.36 KB
Pass Ambil Data User Berhasil	Pass				
GET Team	GET	https://linkmatchsttnfapi.herokuapp.com/api/team	200 OK	7405 ms	2.985 KB
Pass Ambil Data Team Berhasil	Pass				
GET Project	GET	https://linkmatchsttnfapi.herokuapp.com/api/project	200 OK	6351 ms	2.726 KB
Pass Ambil Data Project Berhasil	Pass				
GET Sprint	GET	https://linkmatchsttnfapi.herokuapp.com/api/sprint	200 OK	9263 ms	4.925 KB
Pass Ambil Data Sprint Berhasil	Pass				

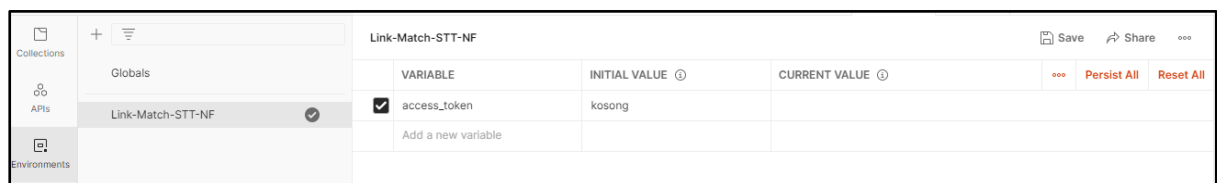
Gambar 7. *Status Test Automation*

Mungkin itu saja terkait status pengujiannya, akan tetapi ada satu hal lagi yang perlu diperhatikan ketika menggunakan *auth* yaitu penggunaan tokennya yang sebelumnya kita lakukan manual harus diubah menjadi dinamis, karena jika tidak maka token akan berubah dan API ambil data tidak dapat diakses disebabkan token yang berganti seperti gambar berikut.



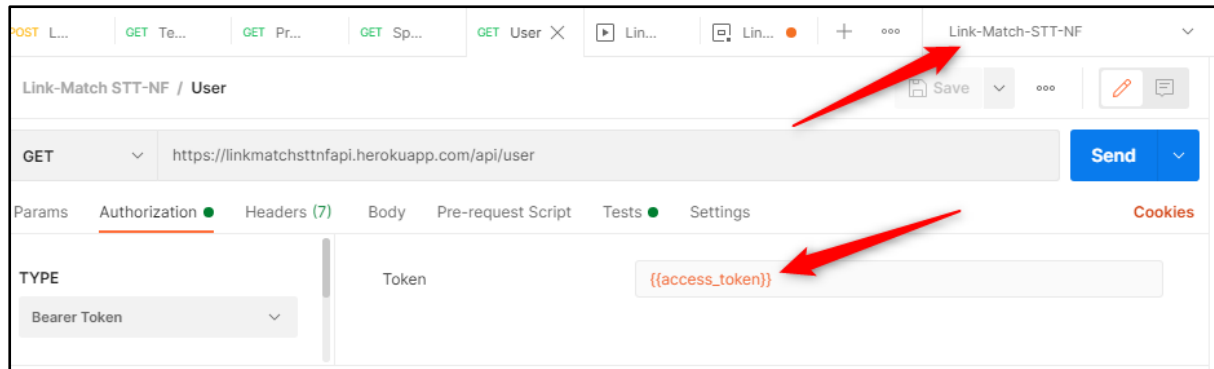
Gambar 8. Gagal Login

Hal tersebut terjadi karena kita sebelumnya mengatur token secara statis bukan dinamis, untuk merubahnya menjadi dinamis kita dapat menggunakan fitur yang telah disediakan oleh Postman yaitu *Environment*. *Environment* memungkinkan kita agar menyimpan nilai dalam sebuah *variable* sehingga dapat kita gunakan dalam menyimpan dan mengambil suatu nilai, cara penerapannya seperti pada gambar berikut.



Gambar 9. Environment

Setelah *environment* dibuat dan *variable* telah dideklarasikan, maka pada API ambil data sebelumnya di bagian *Bearer Token*, cukup memasukkan *variable* yang dituliskan dalam *environment* yaitu *access\_token* dan mengaktifkan *environment* tersebut seperti gambar berikut.



Gambar 10. Deklarasi *Variable* dan Mengaktifkan *Environment*

Hasil yang diberikan akan sama seperti pada gambar 7.

## Penutup

Alhamdulillah penerapan *test automation* untuk web telah berhasil dilakukan, hasil yang didapatkan dari makalah ini adalah ketika akan melakukan suatu pengujian pada sistem, kita tidak perlu melakukannya secara manual terus menerus karena akan membuang waktu dan tenaga. Salah satu solusi yang ditawarkan ialah menjadikan pengujian tersebut menjadi otomatis sehingga dapat mempermudah dalam melihat hasil yang akan didapatkan apakah terjadi kesalahan ataupun berhasil.

## Daftar Pustaka

- <https://www.smashingmagazine.com/2020/09/automate-api-testing-postman/>
- <https://www.youtube.com/watch?v=z0MimkXIVe8>