Sumeet Haldipur
TE COMPS
2019130018
Batch A

# EXPERIMENT 7

## Aim:

Design a secured Web Application using web-token.

## Requirements:

Django, Web Browser(Chrome), API keys of respective services

## Problem Statement:

Olympic management system is an app which targets to provide various sports event organizers a platform to promote their events and also to give the general users/participants information about the sports events in which they're interested

## Theory:

**What is JSON Web Token?**
JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA.

Although JWTs can be encrypted to also provide secrecy between parties, we will focus on signed tokens. Signed tokens can verify the integrity of the claims contained within it, while encrypted tokens hide those claims from other parties. When tokens are signed using public/private key pairs, the signature also certifies that only the party holding the private key is the one that signed it.

**When should you use JSON Web Tokens?**
Here are some scenarios where JSON Web Tokens are useful:

Authorization: This is the most common scenario for using JWT. Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. Single Sign On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used across different domains.

Information Exchange: JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed—for example, using public/private key pairs—you can be sure the senders are who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn't been tampered with.

### What is the JSON Web Token structure?

In its compact form, JSON Web Tokens consist of three parts separated by dots (.), which are:

- Header
- Payload
- Signature

### Header

The header typically consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA.

### Payload

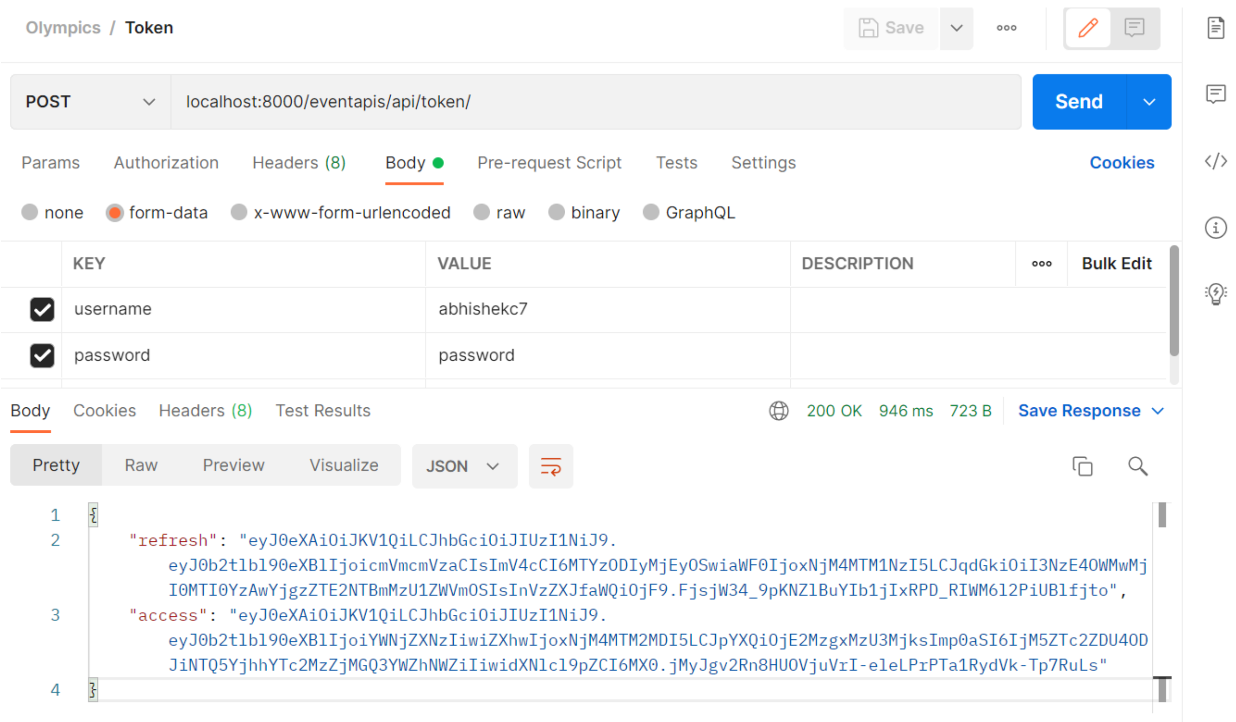
The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional data. There are three types of claims: registered, public, and private claims.

### Signature

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.

The signature is used to verify the message wasn't changed along the way, and, in the case of tokens signed with a private key, it can also verify that the sender of the JWT is who it says it is.

## Implementation:



Olympics / Token

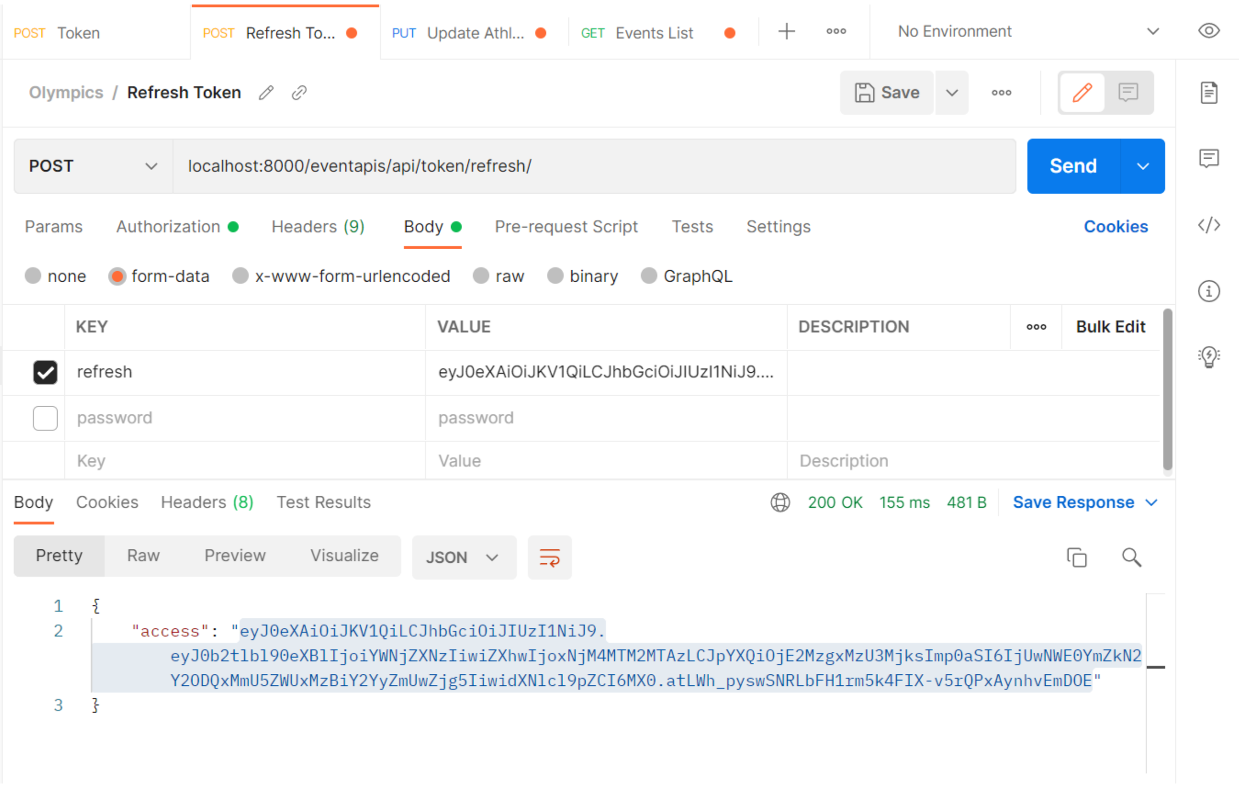
POST localhost:8000/eventapis/api/token/

○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

| | KEY | VALUE | DESCRIPTION |
|---|---|---|---|
| ☑ | username | abhishekc7 | |
| ☑ | password | password | |

200 OK 946 ms 723 B

```
{
    "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
        eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTYzODIyMjEyOSwiaWF0IjoxNjM4MTM1NzI5LCJqdGkiOiI3NzE4OWMwMj
        I0MTI0YzAwYjgzZTE2NTBmMzU1ZWVmOSIsInVzZXJfaWQiOjF9.FjsjW34_9pKNZlBuYIb1jIxRPD_RIWM6l2PiUBlfjto",
    "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
        eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjM4MTM2MDI5LCJpYXQiOjE2MzgxMzU3MjksImp0aSI6IjM5ZTc2ZDU4OD
        JiNTQ5YjhhYTc2MzZjMGQ3YWZhNWZiIiwidXNlcl9pZCI6MX0.jMyJgv2Rn8HUOVjuVrI-eleLPrPTa1RydVk-Tp7RuLs"
}
```



Olympics / Refresh Token

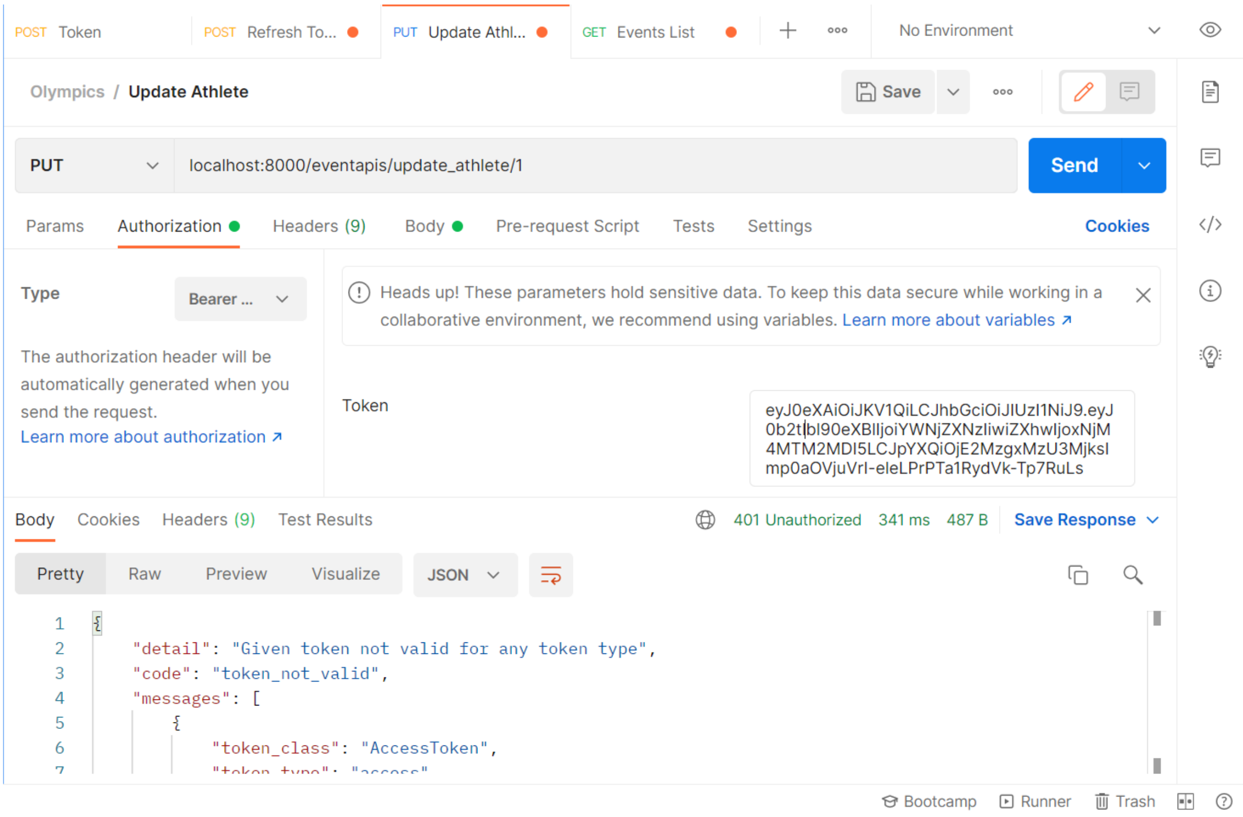
POST localhost:8000/eventapis/api/token/refresh/

○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary ○ GraphQL

| | KEY | VALUE | DESCRIPTION |
|---|---|---|---|
| ☑ | refresh | eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.... | |
| □ | password | password | |
| | Key | Value | Description |

Body — 200 OK 155 ms 481 B

```
{
    "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
        eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjM4MTM2MTAzLCJpYXQiOjE2MzgxMzU3MjksImp0aSI6IjUwNWE0YmZkN2
        Y2ODQxMmU5ZWUxMzBiY2YyZmUwZjg5IiwidXNlcl9pZCI6MX0.atLWh_pyswSNRLbFH1rm5k4FIX-v5rQPxAynhvEmDOE"
}
```



Olympics / Update Athlete

PUT localhost:8000/eventapis/update_athlete/1

Type: Bearer ...

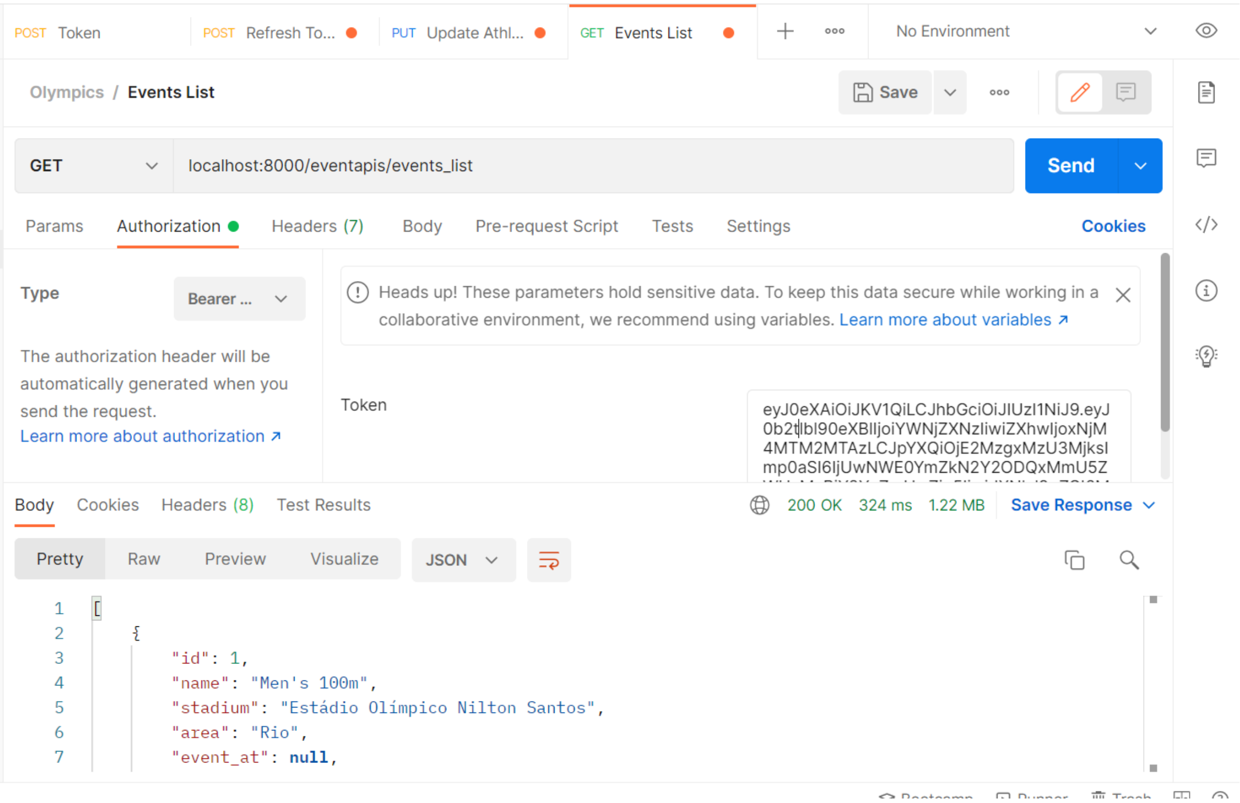
The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjM4MTM2MDI5LCJpYXQiOjE2MzgxMzU3MjksImp0aOVjuVrl-eleLPrPTa1RydVk-Tp7RuLs

Body — 401 Unauthorized 341 ms 487 B

```
{
    "detail": "Given token not valid for any token type",
    "code": "token_not_valid",
    "messages": [
        {
            "token_class": "AccessToken",
            "token_type": "access"
```



GET localhost:8000/eventapis/events_list

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Token

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ
0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjM
4MTM2MTAzLCJpYXQiOjE2MzgxMzU3MjksI
mp0aSI6IjUwNWE0YmZkN2Y2ODQxMmU5Z
```

200 OK 324 ms 1.22 MB

```
[
    {
        "id": 1,
        "name": "Men's 100m",
        "stadium": "Estádio Olímpico Nilton Santos",
        "area": "Rio",
        "event_at": null,
```

## Conclusion:

From the above experiment, I have learned the following:

- Knowledge about web token authentication.
- Hands on experience with web tokens for the given problem statement.

## References:

1. https://jwt.io/introduction
2. https://auth0.com/learn/json-web-tokens/