# JWT authentication – sign in and sign up page – drf and vueJs

## Initial setup from you-tube video:

Video link : <https://www.youtube.com/watch?v=PUzgZrS_piQ&list=PLlameCF3cMEvj7fV-Szcaz_ve303e4AxX&index=1>

=> Django/DRF Backend

-setup

install django

install djangorestframework

1. create project-- django-admin startproject auth

2. create app-- django-admin startapp users

3. settings.py (add)

INSTALLED\_APPS = [

'rest\_framework',

'users', ]

4. users/models.py ---------------

-create users model (db)

-creating user model is different from creating other models in django

from django.contrib.auth.models import AbstractUser

# Create your models here.

class User(AbstractUser):

name = models.CharField(max\_length=100)

email = models.CharField(max\_length=200, unique=True)

password = models.CharField(max\_length=100)

username = None

USERNAME\_FIELD = 'email'

REQUIRED\_FIELDS = [] #empty

5.settings.py -------------------

specify the user model - AUTH\_USER\_MODEL = 'users.User'

6. serializers.py

7. views.py

8. urls.py

=> code logic for creating register function (sign up)

=> code logic for login function (sign in)

## create JWT Tokens

A \*\*JWT (JSON Web Token)\*\* is a way to securely send information between two parties (like a server and a client) as a compact, digitally signed token.

### In simple terms:

- \*\*Think of it as an ID card\*\*: When you log into a website or an app, the server gives you a JWT, just like how a security guard might give you an ID badge.

- \*\*You show the token to prove your identity\*\*: Every time you make a request (like asking for more information from the server), you send this token. The server checks the token to confirm you are who you say you are.

- \*\*It’s secure and cannot be easily tampered with\*\*: The token is "signed" so that if anyone tries to change it, the server will know, and it won't accept it.

### Structure of a JWT:

A JWT has three parts:

1. \*\*Header\*\*: Contains metadata about the token (e.g., type of token and the signing algorithm).

2. \*\*Payload\*\*: Contains the data (like user info, permissions, etc.).

3. \*\*Signature\*\*: Ensures the token wasn’t altered.

When you log out or your token expires, the server won't accept your token anymore, just like when your ID card gets deactivated.

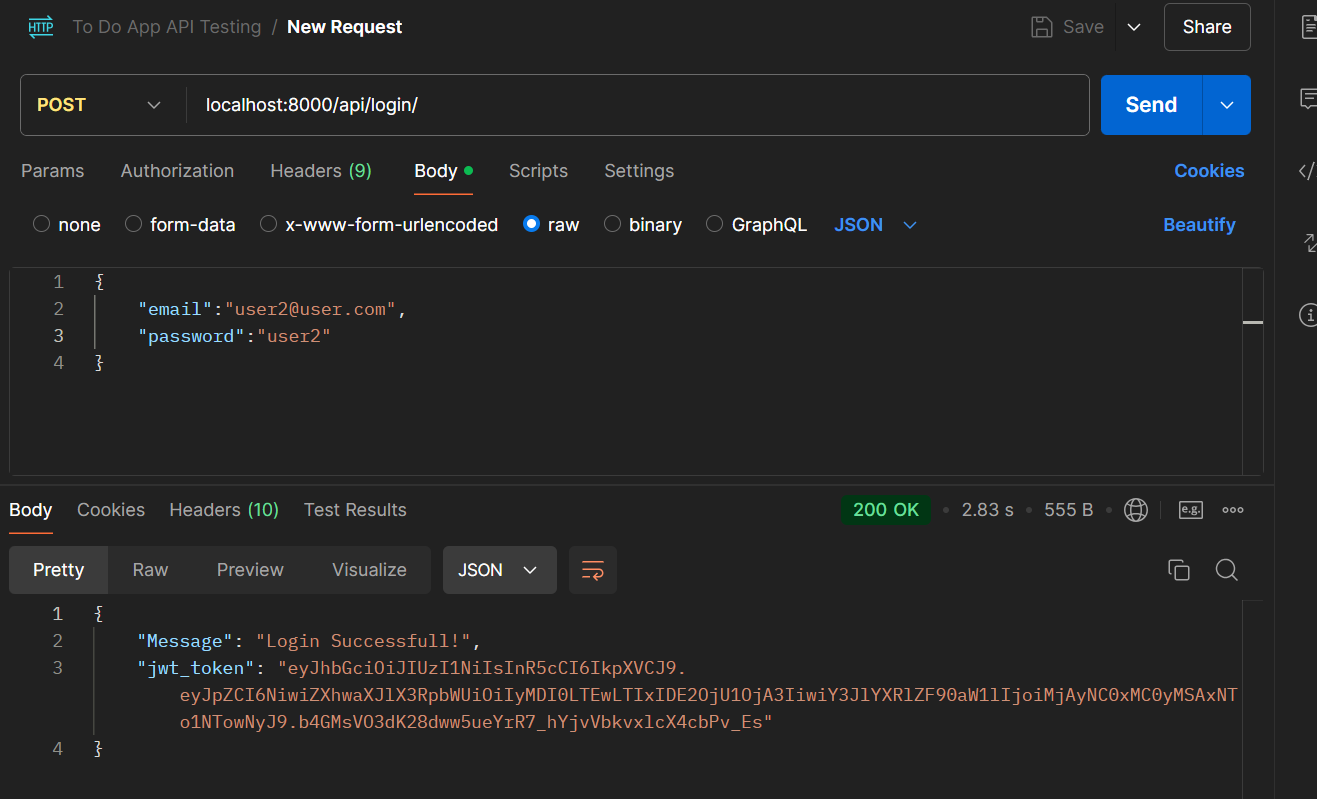
* install a package

python3 –m pip install PyJWT

* Import it in users/Views.py

Import jwt

* Code for it
* Output:



* Install other package

python3 –m pip install django-cors-headers

* Register corsheader in django/settings.py/

installed\_apps – “corsheaders”

middleware – “corsheaders.middleware.CorsMiddleware”

add these variables at end of file :

CORS\_ORIGIN\_ALLOW\_ALL = True (used to access the different frontend ports)

CORS\_ALLOW\_CREDENTIALS = True (used to give access for frontend to GET THOSE Cookies )

### CORSHEADER:

\*\*CORS (Cross-Origin Resource Sharing)\*\* is a security feature built into web browsers that blocks web pages from making requests to a different domain than the one that served the web page.

In simple terms:

- \*\*Without CORS\*\*: If your website (example.com) tries to make a request (like an API call) to another domain (api.example.com), the browser would block it for security reasons.

- \*\*With CORS Headers\*\*: You can tell the browser it's safe to allow these requests by setting special headers on the server side, allowing communication between different domains.

### \*\*Why do you need `django-cors-headers`?\*\*

When you're developing a frontend (like React, Vue.js, etc.) on one server (localhost:3000) and a backend (Django API) on another server (localhost:8000), CORS will block requests from the frontend to the backend.

The `django-cors-headers` package allows you to easily configure your Django server to accept requests from different domains, avoiding the browser's CORS block.

### Example:

Let's say your frontend is at `http://localhost:3000`, and your Django backend is at `http://localhost:8000`. Without CORS headers, the browser would block the frontend from making requests to the backend. But by using `django-cors-headers`, you can tell the backend to allow requests from `http://localhost:3000`.

---

So in simple terms, `django-cors-headers` allows your frontend and backend (or different websites) to communicate safely when they are hosted on different domains.

## Next steps:

* Function for getting the authenticated user using the cookies
* Function to log out the user --- just by deleting the cookies

# Topic 1