**Overview**

Django REST Framework is a powerful and flexible solution for building Web APIs. This library will help you get started with registering users and authenticating them with JSON Web Tokens in your Django project

**Packages/dependencies**

* **Django**
* **Django REST Framework:** After setting up your django project, you’d need to set up the REST framework in order to start working with Web APIs. You can install this by running `python3 -m pip install djangorestframework`
* **PyJWT:** Enables you work with JSON Web Tokens in your Django project. Install with `python3 -m pip install pyJWT`

**Code snippets**

<APP\_NAME>/models.py

from django.contrib.auth.models import AbstractUser

class User(AbstractUser):

    name = models.CharField(max\_length=255)

    email = models.CharField(max\_length=255, unique=True)

    password = models.CharField(max\_length=255)

    username = None

    USERNAME\_FIELD = 'email'

    REQUIRED\_FIELDS = []

<APP\_NAME>/serializers.py

from rest\_framework import serializers

from .models import User

class UserSerializer(serializers.modelSerializer):

    class Meta:

        model = User

        fields = ['id', 'name', 'email', 'password']

        extra\_kwargs = {

            'password': {'write\_only': True}

        }

    def create(self, validated\_data):

        password = validated\_data.pop('password', None)

        instance = self.Meta.model(\*\*validated\_data)

        if password is not None:

            instance.set\_password(password)

        instance.save()

        return instance

<APP\_NAME>/views.py

from rest\_framework.views import APIView

from rest\_framework.response import Response

from rest\_framework.exceptions import AuthenticationFailed

from .serializers import UserSerializer

from .models import User

class RegisterView(APIView):

    def post(self, request):

        serializer = UserSerializer(data=request)

        serializer.is\_valid(raise\_exception=True)

        serializer.save()

        return Response(serializer.data)

**Examples**

Time is precious as a developer, and we want you to get started building that next idea FAST!

To get started using the library:

1. Install Django and Django REST framework then start a project and app
2. Add "rest\_framework" and "<APP\_NAME>" to <PROJECT\_NAME>/settings.py under INSTALLED\_APPS
3. Head to <APP\_NAME>/models.py to create a user model
4. Head to <PROJECT\_NAME>/settings.py and set a new variable => AUTH\_USER\_MODEL = "<APP\_NAME>.User"
5. Run `python3 manage.py makemigrations` and `python3 manage.py migrate` from the project directory to make migrations
6. Include the "/api" route in the project's urls.py and the "/register" route in the app's urls.py

After that is done, you can test your endpoint `localhost:8000/api/register` and **voila!**

**Overview**

Django REST Framework is a powerful and flexible solution for building Web APIs. This library will help you get started with logging in users and authenticating them with JSON Web Tokens in your Django project

**Packages/dependencies**

* **Django**
* **Django REST Framework:** After setting up your django project, you’d need to set up the REST framework in order to start working with Web APIs. You can install this by running `python3 -m pip install djangorestframework`
* **PyJWT:** Enables you work with JSON Web Tokens in your Django project. Install with `python3 -m pip install pyJWT`

**Code snippets**

<APP\_NAME>/models.py

from django.contrib.auth.models import AbstractUser

class User(AbstractUser):

    name = models.CharField(max\_length=255)

    email = models.CharField(max\_length=255, unique=True)

    password = models.CharField(max\_length=255)

    username = None

    USERNAME\_FIELD = 'email'

    REQUIRED\_FIELDS = []

<APP\_NAME>/serializers.py

from rest\_framework import serializers

from .models import User

class UserSerializer(serializers.modelSerializer):

    class Meta:

        model = User

        fields = ['id', 'name', 'email', 'password']

        extra\_kwargs = {

            'password': {'write\_only': True}

        }

    def create(self, validated\_data):

        password = validated\_data.pop('password', None)

        instance = self.Meta.model(\*\*validated\_data)

        if password is not None:

            instance.set\_password(password)

        instance.save()

        return instance

<APP\_NAME>/views.py

from rest\_framework.views import APIView

from rest\_framework.response import Response

from rest\_framework.exceptions import AuthenticationFailed

from .serializers import UserSerializer

from .models import User

import jwt

import datetime

class LoginView(APIView):

    def post(self, request):

        email = request.data['email']

        password = request.data['password']

        user = User.objects.filter(email=email).first()

        if user is None:

            raise AuthenticationFailed("User not found")

        if not user.check\_password(password):

            raise AuthenticationFailed("Incorrect password")

        payload = {

            'id': user.id,

            'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=60),

            'iat': datetime.datetime.utcnow()

        }

*# ensure to hide secret in production*

        token = jwt.encode(payload, 'SUP3R\_S3CUR3\_SECRET',

                           algorithm='HS256').decode('utf-8')

        response = Response()

        response.set\_cookie(key='jwt', value=token, httponly=True)

        response.data = {

            "jwt": token

        }

        return response

*# decoding the token and returning to user to the frontend*

class UserView(APIView):

    def get(self, request):

        token = request.COOKIES.get('jwt')

        if not token:

            raise AuthenticationFailed("Unauthenticated")

        try:

            payload = jwt.decode(

                token, 'SUP3R\_S3CUR3\_SECRET', algorithm=['HS256'])

        except jwt.ExpiredSignatureError:

            raise AuthenticationFailed("Unauthenticated")

        user = User.objects.filter(id=payload['id']).first()

        serializer = UserSerializer(user)

        return Response(serializer.data)

**Examples**

Time is precious as a developer, and we want you to get started building that next idea FAST!

To get started using the library:

1. Install Django and Django REST framework then start a project and app
2. Add "rest\_framework" and "<APP\_NAME>" to <PROJECT\_NAME>/settings.py under INSTALLED\_APPS
3. Head to <APP\_NAME>/models.py to create a user model
4. Head to <PROJECT\_NAME>/settings.py and set a new variable => AUTH\_USER\_MODEL = "<APP\_NAME>.User"
5. Run `python3 manage.py makemigrations` and `python3 manage.py migrate` from the project directory to make migrations
6. Include the "/api" route in the project's urls.py and the "/login" route in the app's urls.py
7. Run `python3 -m pip install pyJWT` to install the JWT package
8. Run `python3 -m pip install django-cors-headers` and add it to your project's installed apps as "corsheaders" in settings.py
9. Also add the "corsheaders.middleware.CorsMiddleware" to project's middleware list
10. Add two new variables => `CORS\_ORIGIN\_ALLOW\_ALL = True` and `CORS\_ALLOW\_CREDENTIALS = True` in settings.py

After that is done, you can test your endpoint `localhost:8000/api/login` and **voila!**

**Overview**

Django REST Framework is a powerful and flexible solution for building Web APIs. This library will help you get started with logging out users and de-authenticating them using JSON Web Tokens and Cookies in your Django project

**Packages/dependencies**

* **Django**
* **Django REST Framework:** After setting up your django project, you’d need to set up the REST framework in order to start working with Web APIs. You can install this by running `python3 -m pip install djangorestframework`
* **PyJWT:** Enables you work with JSON Web Tokens in your Django project. Install with `python3 -m pip install pyJWT`

**Code snippets**

<APP\_NAME>/models.py

from django.contrib.auth.models import AbstractUser

class User(AbstractUser):

    name = models.CharField(max\_length=255)

    email = models.CharField(max\_length=255, unique=True)

    password = models.CharField(max\_length=255)

    username = None

    USERNAME\_FIELD = 'email'

    REQUIRED\_FIELDS = []

<APP\_NAME>/serializers.py

from rest\_framework import serializers

from .models import User

class UserSerializer(serializers.modelSerializer):

    class Meta:

        model = User

        fields = ['id', 'name', 'email', 'password']

        extra\_kwargs = {

            'password': {'write\_only': True}

        }

    def create(self, validated\_data):

        password = validated\_data.pop('password', None)

        instance = self.Meta.model(\*\*validated\_data)

        if password is not None:

            instance.set\_password(password)

        instance.save()

        return instance

<APP\_NAME>/views.py

from rest\_framework.views import APIView

from rest\_framework.response import Response

from rest\_framework.exceptions import AuthenticationFailed

from .serializers import UserSerializer

from .models import User

class LogoutView(APIView):

    def post(self, request):

        response = Response()

        response.delete\_cookie('jwt')

        response.data = {

            'message': 'Logout successful'

        }

        return response

**Examples**

Time is precious as a developer, and we want you to get started building that next idea FAST!

To get started using the library:

1. Install Django and Django REST framework then start a project and app
2. Add "rest\_framework" and "<APP\_NAME>" to <PROJECT\_NAME>/settings.py under INSTALLED\_APPS
3. Head to <APP\_NAME>/models.py to create a user model
4. Head to <PROJECT\_NAME>/settings.py and set a new variable => AUTH\_USER\_MODEL = "<APP\_NAME>.User"
5. Run `python3 manage.py makemigrations` and `python3 manage.py migrate` from the project directory to make migrations
6. Include the "/api" route in the project's urls.py and the "/logout" route in the app's urls.py
7. Run `python3 -m pip install pyJWT` to install the JWT package
8. Run `python3 -m pip install django-cors-headers` and add it to your project's installed apps as "corsheaders" in settings.py
9. Also add the "corsheaders.middleware.CorsMiddleware" to project's middleware list
10. Add two new variables => `CORS\_ORIGIN\_ALLOW\_ALL = True` and `CORS\_ALLOW\_CREDENTIALS = True` in settings.py
11. Add the logout view to urls.py

After that is done, you can test your endpoint `localhost:8000/api/logout` and **voila!**