COMPILED BY :JUDAH T CHISARE
:JUNIOR FULL STACK DEVELOPER /JUNIOR DATA ENGINEER
CONTACT :0785430774/0777222531
:jaycea247@gmail.com/ Judah.Chisare@cassavasmartech.co.zw

JWT AUTHENTICATED DJANGO VIEWS

NB: Please note that to use jwt authenticated views you must try and use an abstract User model.For you to implement an Abstract User model,it has to be the firsy model you write and migrate in the models.py.

1 . Abstract user model

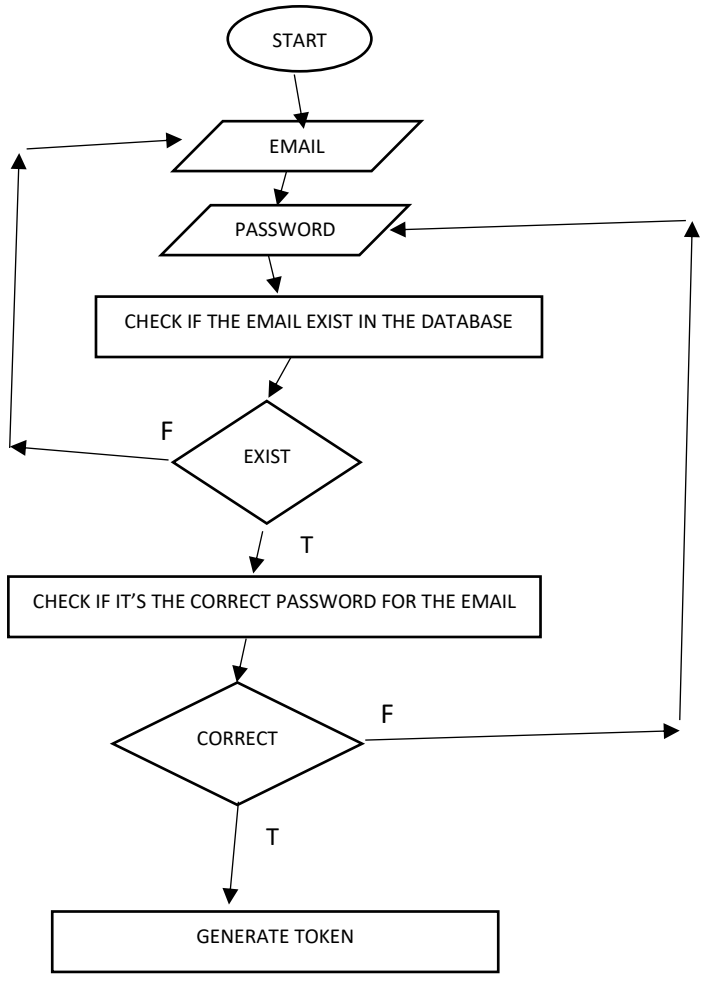
```
from django.db import models
from django.contrib.auth.models import AbstractUser

# Create your models here.
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 = []
```

2.VALIDATING THE EMAIL AND PASSWORD

The process of validating the credentials is presented by the following flowchart



3. GENERATING THE TOKEN

```
import jwt, datetime
```

A)**Payload**

Create a payload which is a dictionary of three elements which are the user id,expiry time and the logging in time.

```
        payload = {
            'id':user.id,
            'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=10),
            'iat': datetime.datetime.utcnow()
        }
```

B) **TOKEN**

now create a token variable which encodes three elements which are the payload,'secret' and the algorithm with jwt.

```
token = jwt.encode(payload, 'secret', algorithm='HS256')
```

c)**COOKIE**

Now set up the cookie which containts a dictionary of three elements

1: key : the cookie name,

2: value : the actual token,

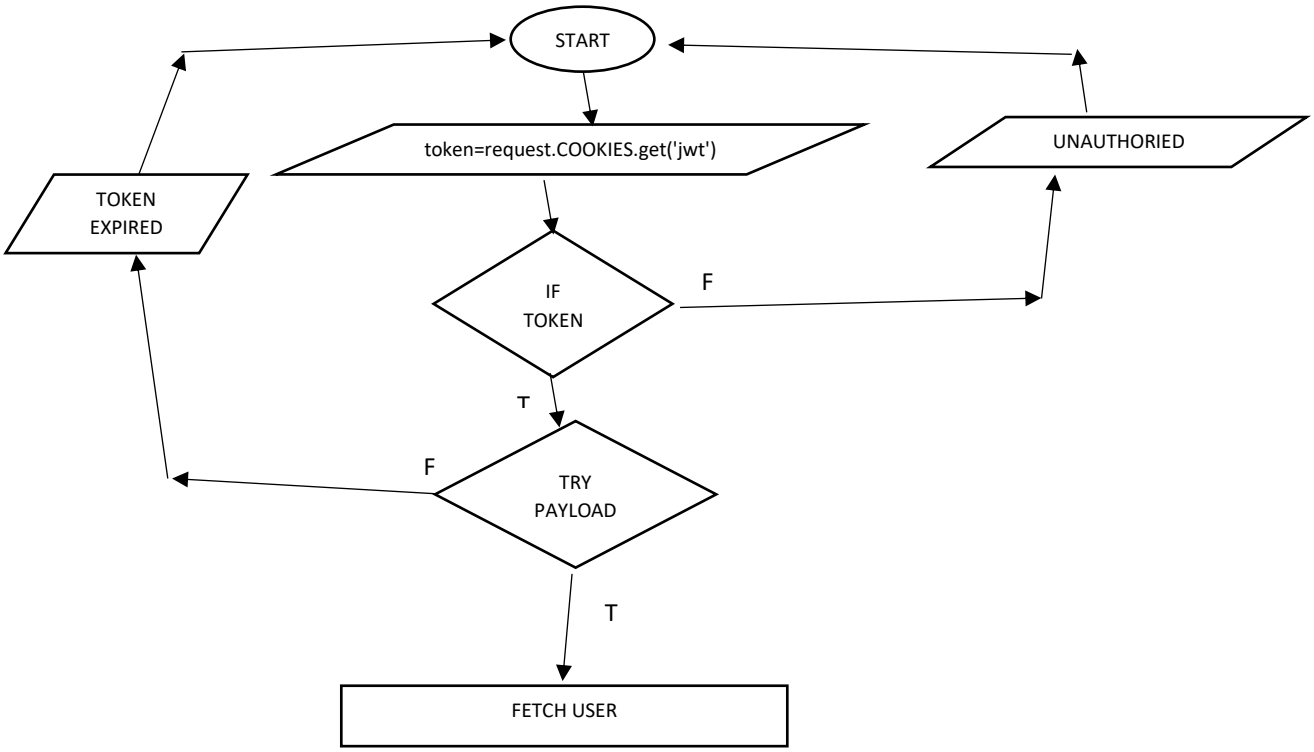
3: HOnly : True

Main function is the .set_cookie(key:'',value:'',httponly:True)

```
    response = Response()
         response.set_cookie(key='jwt',value=token ,httponly=True)
         response.data = {
             'jwt':token
         }
         return response
```

4 .PORTAL WITH THE LOGED IN USER

Note that the following authaurisation algorithm is the one which applies in every function



NB : Note that we have to decode the encoded payload to extract the token using

```
payload =jwt.decode(token, 'secret', algorithms=['HS256'])
```

```python
class UserView(APIView):
    def get(self,request):

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

        if not token:
            raise AuthenticationFailed("unauthorised")

        try:
            payload =jwt.decode(token, 'secret', algorithms=['HS256'])
        except jwt.ExpiredSignatureError:
            raise AuthenticationFailed("session expired")

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

        serializer=UserSerializer(user)
        return Response(serializer.data)
```

5. THE UPDATE AND DELETE

At this point inside the main Main class we will create a universal function which will be called inside the whole class to reduce redundancy.

```
class Update(APIView):
      def get_object(self,request):
            try:
                token=request.COOKIES.get('jwt')

                if not token:
                    raise AuthenticationFailed("unauthorised")

                try:
                    payload =jwt.decode(token, 'secret', algorithms=['HS256'])
                except jwt.ExpiredSignatureError:
                    raise AuthenticationFailed("session expired")

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

                return user

            except User.DoesNotExist:
                return Response("wakadhakwa",status=status.HTTP_204_NO_CONTENT
)

      def get(self,request):
            obj=self.get_object(request)
            serializer=UserSerializer(obj)
            return Response(serializer.data)


      def put(self,request):

        obj=self.get_object(request)
        serializer=UserSerializer(obj,data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response("corrupted data",status=status.HTTP_204_NO_CONTENT)

      def delete(self,request):
        all=self.get_object(request)
        all.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

6. LOGOUT

The main concept here is just to delete the cookie

```
class LogoutView(APIView):
    def post(self, request):
        response = Response()
        response.delete_cookie('jwt')
        response.data = {
            'message': 'success'
        }
        return response
```