OFFLINE STORE MANAGEMENT SYSTEM-
AUTHENTICATION MODULE

A Project Submitted by :

Busappagari Shanmukha(Team lead)

Yejju Vamsi Krishna

Ravi Mishra

Siva Shankhar

Venkata Sai Kumar

MAROLIX TECHNOLOGY SOLUTIONS PVT LTD

FEB & 2024

Requirements : Vs code,

Python IDLE,

Postgresql,

Postman tool

Packages & Software :- Python3,

Virtualenv,

Django 4.2,

Django_restframework,

Smtp Configuration,

Postgresql v16,

Documentation

step -1 :

Creating project folder and virtual environment

we need to create a new folder and name it later on we have to open that folder that folder on vs code and next we have to click on terminal, select new terminal and then we have to use the following commands:

*python -- version

we have to check the version of python

*pip -- version

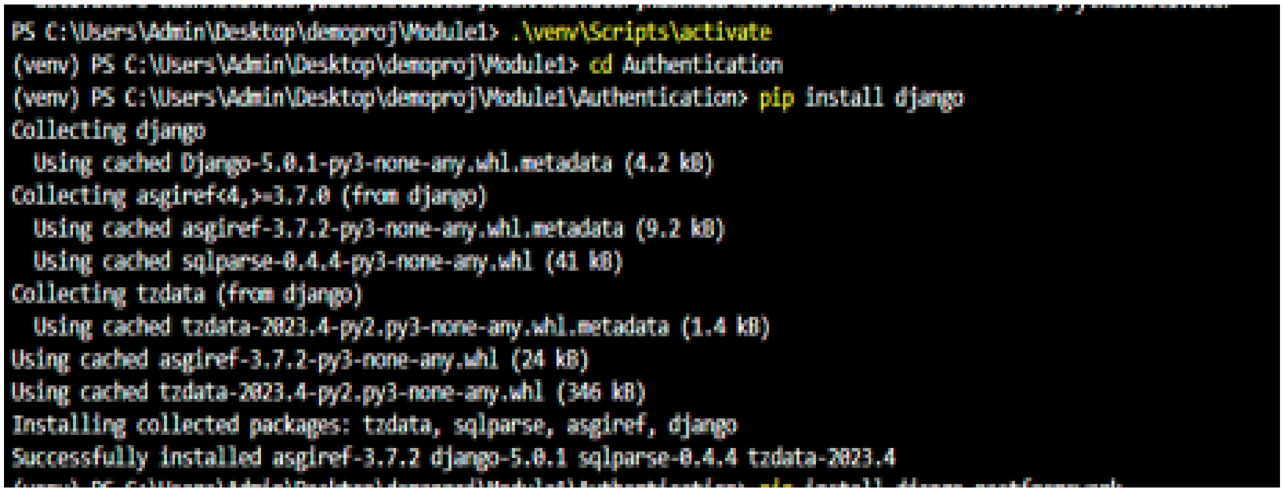
we have to check pip version

By using the below commands we have to activate virtual environment

*pip install virtualenv

*python -m virtualenv venv

*.\venv\Scripts\activate



```
PS C:\Users\Admin\Desktop\demoproj\Module1> .\venv\Scripts\activate
(venv) PS C:\Users\Admin\Desktop\demoproj\Module1> cd Authentication
(venv) PS C:\Users\Admin\Desktop\demoproj\Module1\Authentication> pip install django
Collecting django
  Using cached Django-5.0.1-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.7.0 (from django)
  Using cached asgiref-3.7.2-py3-none-any.whl.metadata (9.2 kB)
  Using cached sqlparse-0.4.4-py3-none-any.whl (41 kB)
Collecting tzdata (from django)
  Using cached tzdata-2023.4-py2.py3-none-any.whl.metadata (1.4 kB)
Using cached asgiref-3.7.2-py3-none-any.whl (24 kB)
Using cached tzdata-2023.4-py2.py3-none-any.whl (346 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.7.2 django-5.0.1 sqlparse-0.4.4 tzdata-2023.4
```

By using this command virtual environment is created successfully.

In this python project we have to run the django server for that we are using django command

*pip install django

*pip install pyscopg2 # to connect with database

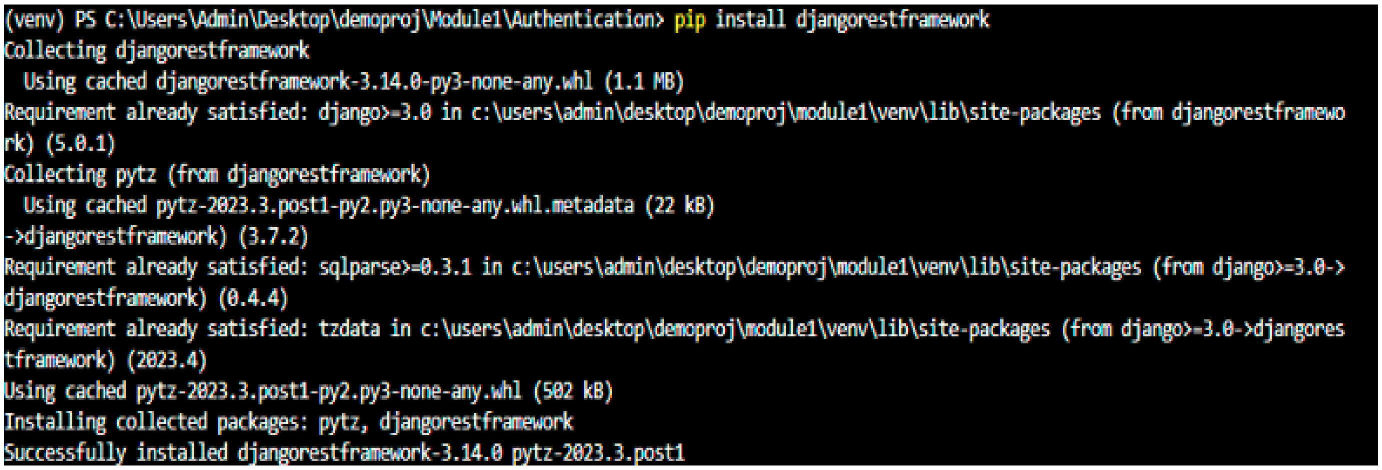
*django-admin startproject firstproject

*cd -\firstproject\

*python manage.py startapp firstapp

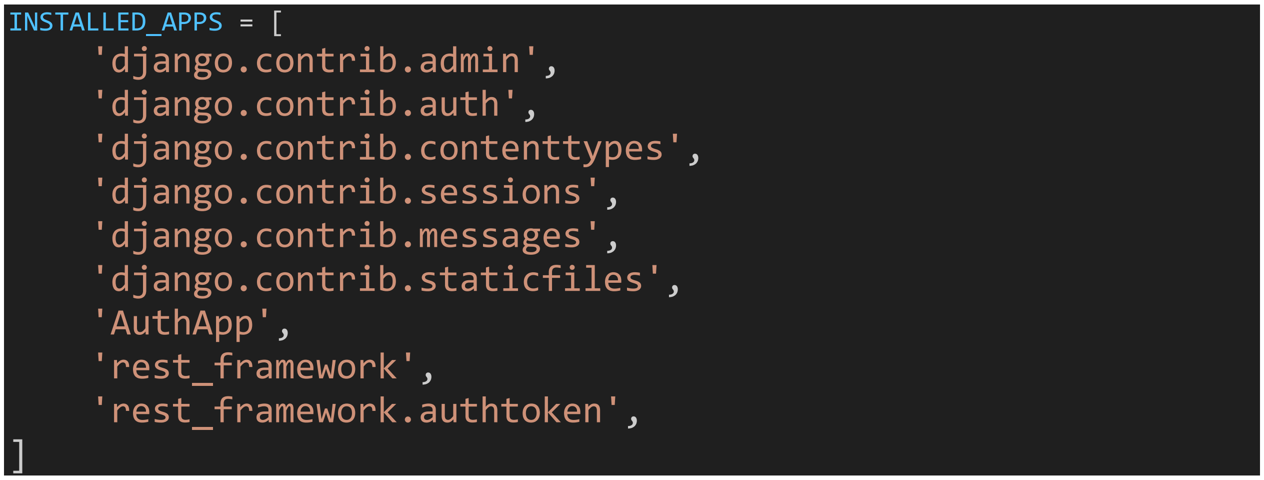
we have to install django rest framework so we are using the below command

*pip install djangorestframework



```
(venv) PS C:\Users\Admin\Desktop\demoproj\Module1\Authentication> pip install djangorestframework
Collecting djangorestframework
  Using cached djangorestframework-3.14.0-py3-none-any.whl (1.1 MB)
Requirement already satisfied: django>=3.0 in c:\users\admin\desktop\demoproj\module1\venv\lib\site-packages (from djangorestframewo
rk) (5.0.1)
Collecting pytz (from djangorestframework)
  Using cached pytz-2023.3.post1-py2.py3-none-any.whl.metadata (22 kB)
->djangorestframework) (3.7.2)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\admin\desktop\demoproj\module1\venv\lib\site-packages (from django>=3.0->
djangorestframework) (0.4.4)
Requirement already satisfied: tzdata in c:\users\admin\desktop\demoproj\module1\venv\lib\site-packages (from django>=3.0->djangores
tframework) (2023.4)
Using cached pytz-2023.3.post1-py2.py3-none-any.whl (502 kB)
Installing collected packages: pytz, djangorestframework
Successfully installed djangorestframework-3.14.0 pytz-2023.3.post1
```

- Go to settings.py file , Under Insatlled Apps
  Add Application name and add
  rest_framework, rest_framework.authtoken



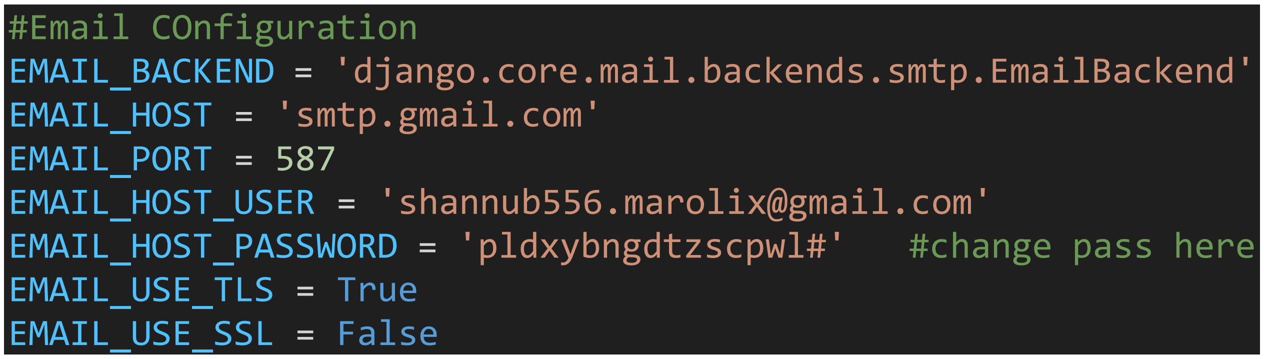
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'AuthApp',
    'rest_framework',
    'rest_framework.authtoken',
]
```

After that go setting.py file and replace DATABASE with below code

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
         'NAME': 'Module_db',
         'USER': 'postgres',
         'PASSWORD': '123',
         'HOST': 'localhost',
         'PORT': '5432'
    }
}
```

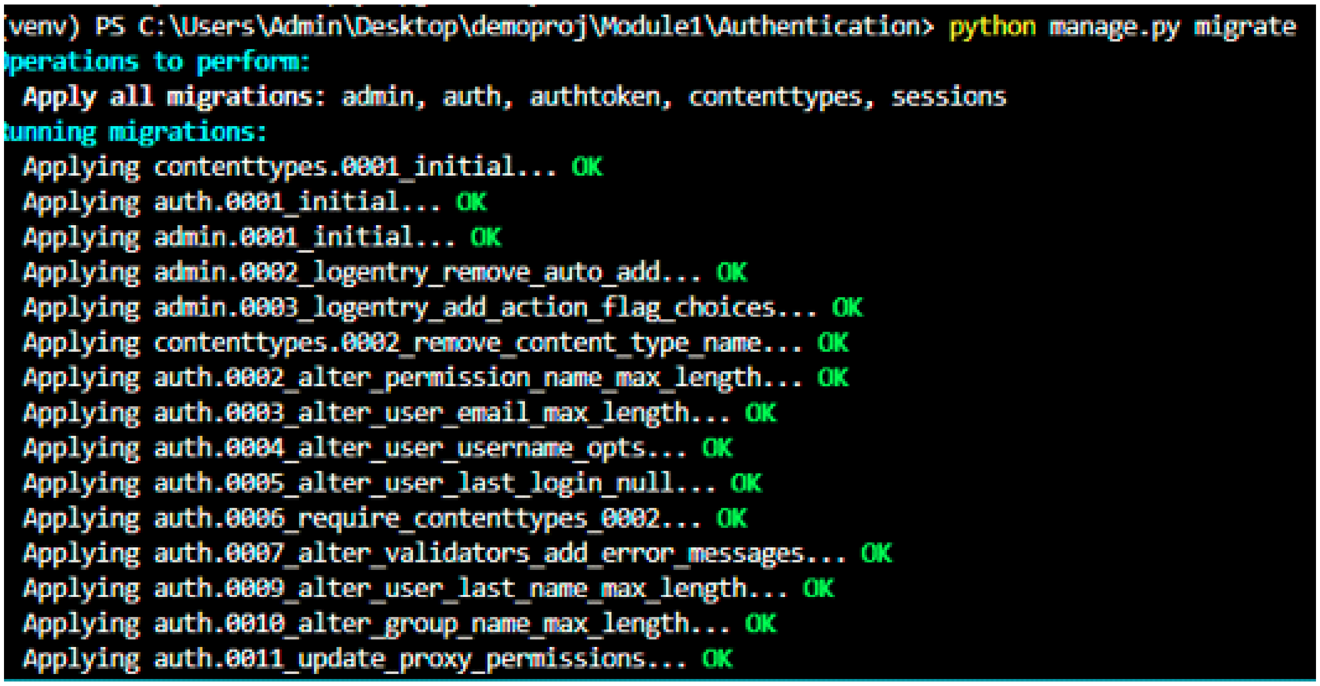
- Go to gmail and create one app for SMTP Configuration , copy that app password

Now , In settings.py file add this at the end of code



```
#Email COnfiguration
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = 'shannub556.marolix@gmail.com'
EMAIL_HOST_PASSWORD = 'pldxybngdtzscpwl#'   #change pass here
EMAIL_USE_TLS = True
EMAIL_USE_SSL = False
```

*python manage.py migrate



```
(venv) PS C:\Users\Admin\Desktop\demoproj\Module1\Authentication> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, authtoken, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
```

*Go to urls.py in project folder and add include with below code

```
path('',include('AuthApp.urls')),
```

*And create one more urls.py in our Application and then write below code

```
from .import views

urlpatterns = [
    path('',views.input),
    path('login/',views.login,name='login'),
    path('reset/',views.reset,name='reset'),
    path('change-password/',
views.change_password,
name='change_password'),
    path('visitors/',views.visitors,name='visitors'),
    path('logout/',views.logout,name='logout'),
]
```

- And then create one more seriailizers file in your Application folder and add below code

```python
from rest_framework import serializers
from django.contrib.auth.models import User
from .models import User_data

class userseralizer(serializers.ModelSerializer):
    class Meta:
        model=User
        fields=['id','username','password','email']


class user_data_seralizer(serializers.ModelSerializer):
    class Meta:
        model=User_data
        fields='__all__'
```

Step-2:

Creating the user

We need to import module User from Django package by using

- ➔ from django.contrib.auth.models import User

And then pass the attributes in that class by using syntax

- ➔ user_details=User.objects.create_user(user name= user_username , password=user_password, email=user_email)
- ➔ user_details.save()   # To save that user

Here ,

(user_username, user_password , user_email) =input data which Is taken from user

- ➔ And for the Email we need to import EmailMessage from Django package by using
- from django.core.mail import EmailMessage

  Emailmessage will take email address to send email and it we can send mail by using smtp configuration

NOTE:

By using saving the user using User Module, it will save the relevant data in database along with the encrypted password , which enhance security for password

And we can get tokens by using this Token inbuilt method

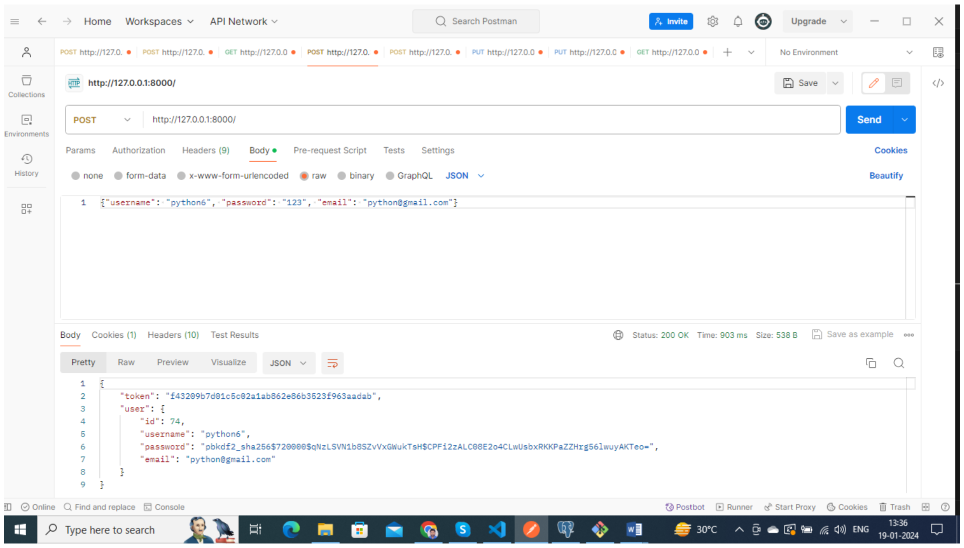
By using this syntax

- ➔ token=Token.objects.create(user=user_details)

```
@api_view(["POST"])                          #csrf token will be
verification will be done here
def input(request):
    if request.method=='POST':
        username=request.data['username']
        password=request.data['password']
        email=request.data['email']
        try:
            user_details=User.objects.create_user(username=u
sername,password=password,email=email)  #Saving the user
            user_details.save()
            token=Token.objects.create(user=user_details)
            serilizer1=user_data_seralizer(data=request.data
)
            if serilizer1.is_valid():
                serilizer1.save()
            email=EmailMessage(
     #syntax to send email
        subject = f'{username}-New user Registered',
        body = f'A new-user with username -{username} and with
a email-{email} was succesfully registered ',
        from_email =
'shannub556.marolix@gmail.com',    #default address(domain)
        #to=[email],                                       #user
email adress(input email from user)
        to = ['bussapagarishannu@gmail.com'],             #user
email adress(Demo mail)
        bcc= ['temporaryb556@gmail.com'])                #admin
adress
            email.send()
        except:
            return Response({"Message" : "username already
exsists try again "})         # if username already exsists
        serilizer=userseralizer(user_details)    #converting
the user details using serilizer
        return Response({'token':token.key, "user":
serilizer.data})          #returning the data
```

Performing Unit Testing :

With the help of Postman tool, we will be performing various test according to input

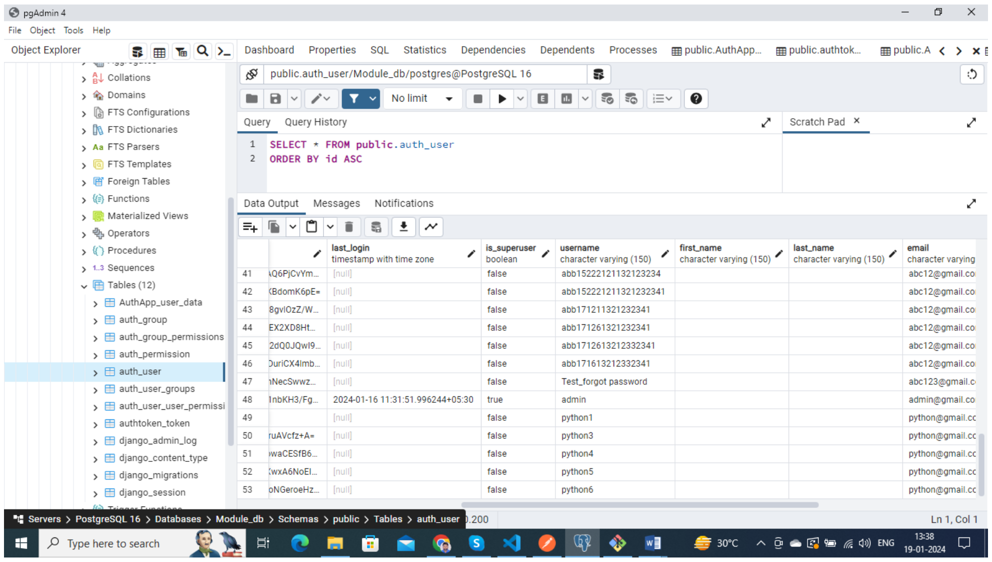


POST http://127.0.0.1:8000/

```
{"username": "python6", "password": "123", "email": "python@gmail.com"}
```

Status: 200 OK Time: 903 ms Size: 538 B

```
{
    "token": "f43209b7d01c5c02a1ab862e86b3523f963aadab",
    "user": {
        "id": 74,
        "username": "python6",
        "password": "pbkdf2_sha256$720000$qNzLSVN1b8SZvVxGWukTsH$CPFi2zALC08E2o4CLwUsbxRKKPaZZHrg56lwuyAKTeo=",
        "email": "python@gmail.com"
    }
}
```



public.auth_user/Module_db/postgres@PostgreSQL 16

```
SELECT * FROM public.auth_user
ORDER BY id ASC
```

		last_login timestamp with time zone	is_superuser boolean	username character varying (150)	first_name character varying (150)	last_name character varying (150)	email character varying
41	AQ6PjCvYm...	[null]	false	abb15222121132123234			abc12@gmail.co
42	KBdomK6pE=	[null]	false	abb152221211321232341			abc12@gmail.co
43	8gvlOzZ/W...	[null]	false	abb171211321232341			abc12@gmail.co
44	EX2XD8Ht...	[null]	false	abb171261321232341			abc12@gmail.co
45	2dQ0JQwI9...	[null]	false	abb1712613212332341			abc12@gmail.co
46	DuriCX4lmb...	[null]	false	abb171613212332341			abc12@gmail.co
47	nNecSwwz...	[null]	false	Test_forgot password			abc123@gmail.c
48	1nbKH3/Fg...	2024-01-16 11:31:51.996244+05:30	true	admin			admin@gmail.co
49		[null]	false	python1			python@gmail.cc
50	ruAVcfz+A=	[null]	false	python3			python@gmail.cc
51	waCESfB6...	[null]	false	python4			python@gmail.cc
52	XwxA6NoEI...	[null]	false	python5			python@gmail.cc
53	oNGeroeHz...	[null]	false	python6			python@gmail.cc

Servers > PostgreSQL 16 > Databases > Module_db > Schemas > public > Tables > auth_user

Login APi:

When user needs to login , user needs to provide username and password

Basically in Django password is encrypted while saving the username and password , to verify that encrypted password   we need to use inbuilt method authenticate method which is available in Django inbuilt package

By using this syntax

- ➔ from django.contrib.auth import authenticate

  If username and password is not valid , then it will return None

And we can get tokens by using this Token inbuilt method orelse if token is not there then it will generate new token

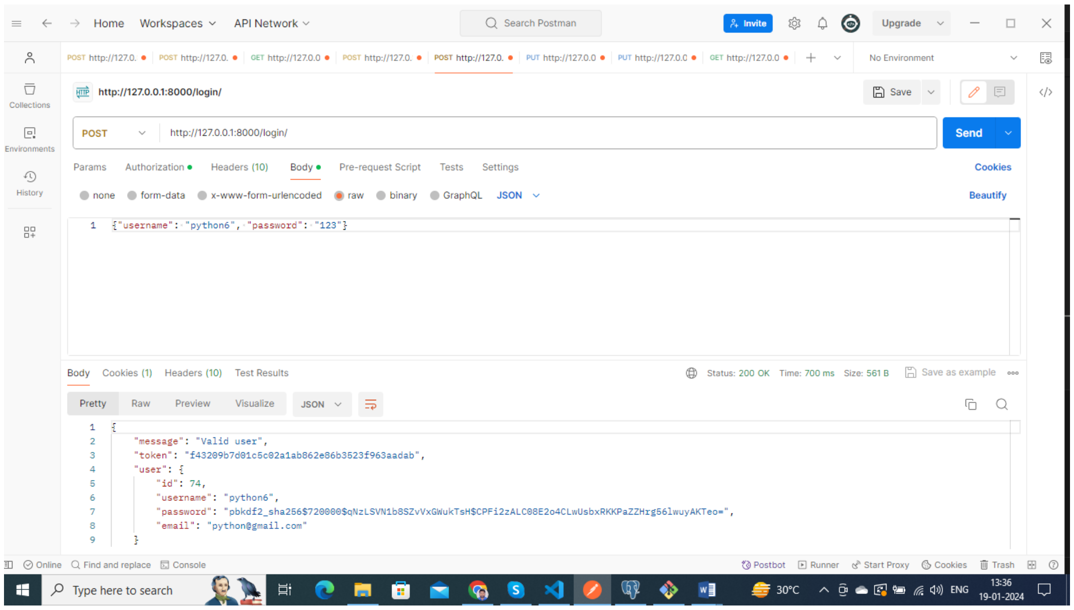
By using this syntax

- ➔ token,value=Token.objects.get_or_create(user=user_details)

```python
def login(request):
    if request.method=="POST":
        username=request.data['username']
        password=request.data['password']
        user_details=authenticate(username=username,password=password)     #if user is not valid it will return None
        if user_details is not None:
            user_details=User.objects.get(username=username)
            token,value=Token.objects.get_or_create(user=user_details)
            serializer=userseralizer(user_details)
            user_data=User_data.objects.get(username=username)
            serializer1=user_data_seralizer(user_data)
            count=serializer1.data['count']
            count=int(count)+1
            User_data.objects.filter(username=username).update(count=count)
            return Response({'message': 'Valid user','token':token.key,'user':serializer.data})
    return Response({'message':'Invalid User'})
```

Performing Unit Testing :

With the help of Postman tool, we will be performing various test according to input

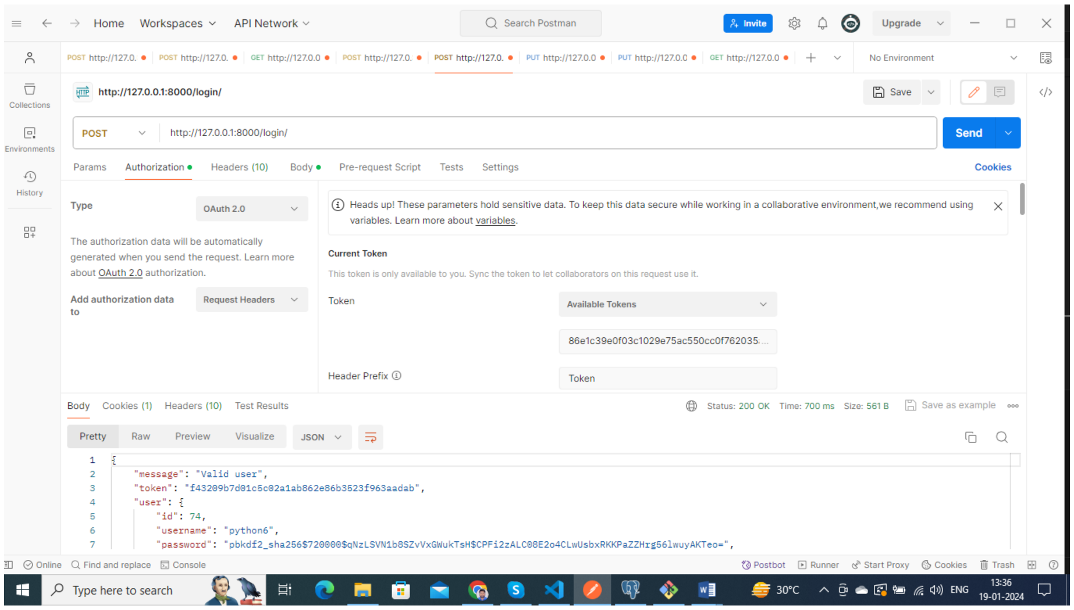


POST http://127.0.0.1:8000/login/

```
{"username": "python6", "password": "123"}
```

Status: 200 OK Time: 700 ms Size: 561 B

```
{
    "message": "Valid user",
    "token": "f43209b7d01c5c02a1ab862e86b3523f963aadab",
    "user": {
        "id": 74,
        "username": "python6",
        "password": "pbkdf2_sha256$720000$qNzLSVN1b8SZvVxGWukTsH$CPFi2zALC08E2o4CLwUsbxRKKPaZZHrg56lwuyAKTeo=",
        "email": "python@gmail.com"
    }
```



POST http://127.0.0.1:8000/login/

Authorization

Type: OAuth 2.0

The authorization data will be automatically generated when you send the request. Learn more about OAuth 2.0 authorization.

Add authorization data to: Request Headers

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.

Current Token

This token is only available to you. Sync the token to let collaborators on this request use it.

Token: Available Tokens

86e1c39e0f03c1029e75ac550cc0f762035...

Header Prefix: Token

Status: 200 OK Time: 700 ms Size: 561 B

```
{
    "message": "Valid user",
    "token": "f43209b7d01c5c02a1ab862e86b3523f963aadab",
    "user": {
        "id": 74,
        "username": "python6",
        "password": "pbkdf2_sha256$720000$qNzLSVN1b8SZvVxGWukTsH$CPFi2zALC08E2o4CLwUsbxRKKPaZZHrg56lwuyAKTeo=",
```

Reset Api:

In this Api, User can able to reset his password with the help of his email, Firstly it will check whether user with that corresponding email exsist or not if exsist then it will take password and password1 , it will verify whether password and password1 are same , if same it will update that old password with given new password

To Update password in django we will be having an inbulit method i.e “set_password”

By using synatx :

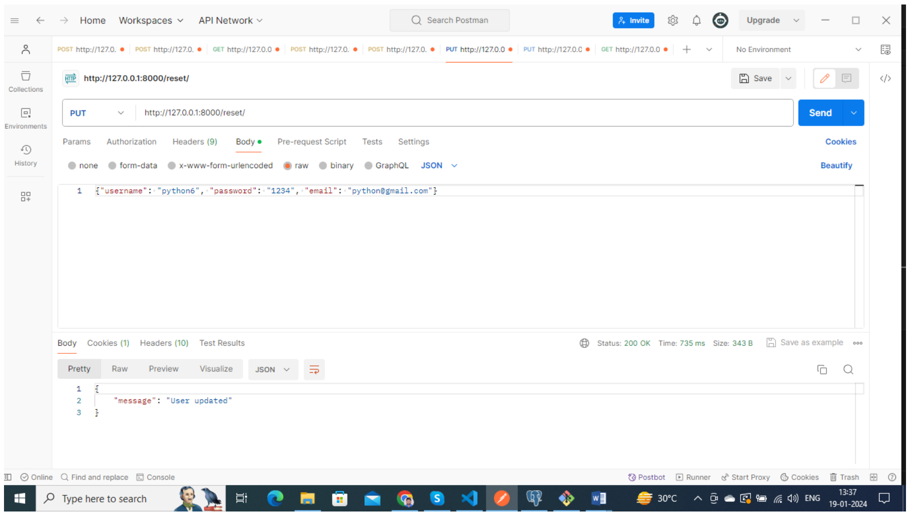
- ➔ user=User.objects.get(username=username)
- ➔ user.set_password(new_password)
  #set_password is inbulit method which will encrpt newpassword
- ➔ user.save()

```
def reset(request):
    if request.method == "PUT":
        username = request.data.get('username')
        new_password = request.data.get('password')
        email = request.data.get('email')
        if not all([username, new_password, email]):
            return Response({'message': "All fields are
required."}, status=status.HTTP_400_BAD_REQUEST)
        try:
            user_details =
User.objects.get(username=username)
            serializer = userseralizer(user_details)
            if email == serializer.data['email']:
                #user, created =
User.objects.update_or_create(username=username,
email=email,defaults={'password': new_password})          #
this method will automatically replace and save paticular feild
, but it won't encript password
                user=User.objects.get(username=username)
                user.set_password(new_password)
                                       #set_password is
inbulit method which will encrpt new password
                user.save()
                                       #it will save encrypted
password
                return Response({'message': "User
updated"})

            else:
                return Response({'message': "Email does not
match."}, status=status.HTTP_400_BAD_REQUEST)
        except User.DoesNotExist:
            return Response({'message': "User not found."},
status=status.HTTP_404_NOT_FOUND)
    return Response(status=status.HTTP_200_OK)
```

Performing Unit Testing :

With the help of Postman tool, we will be performing various test according to input

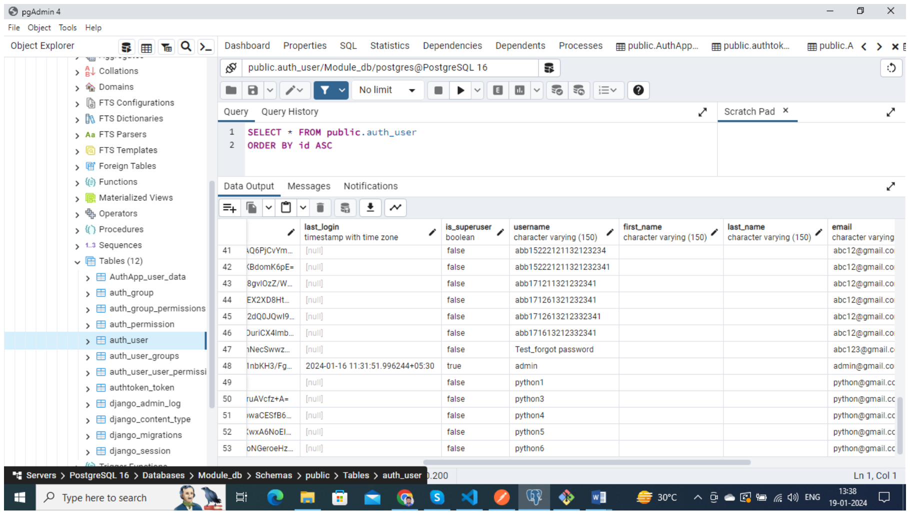


PUT http://127.0.0.1:8000/reset/

```
{"username": "python6", "password": "1234", "email": "python@gmail.com"}
```

Status: 200 OK Time: 735 ms Size: 343 B

```
{
    "message": "User updated"
}
```



```
SELECT * FROM public.auth_user
ORDER BY id ASC
```

		last_login timestamp with time zone	is_superuser boolean	username character varying (150)	first_name character varying (150)	last_name character varying (150)	email character varying
41	AQ6PjCvYm...	[null]	false	abb1522121132123234			abc12@gmail.co
42	KBdomK6pE=	[null]	false	abb152221211321232341			abc12@gmail.co
43	8gvlOzZ/W...	[null]	false	abb171211321232341			abc12@gmail.co
44	EX2XD8Ht...	[null]	false	abb171261321232341			abc12@gmail.co
45	2dQ0JQwI9...	[null]	false	abb1712613212332341			abc12@gmail.co
46	DuriCX4Imb...	[null]	false	abb171613212332341			abc12@gmail.co
47	nNecSwwz...	[null]	false	Test_forgot password			abc123@gmail.c
48	1nbKH3/Fg...	2024-01-16 11:31:51.996244+05:30	true	admin			admin@gmail.co
49		[null]	false	python1			python@gmail.cc
50	ruAVcfz+A=	[null]	false	python3			python@gmail.cc
51	waCESfB6...	[null]	false	python4			python@gmail.cc
52	KwxA6NoEI...	[null]	false	python5			python@gmail.cc
53	oNGeroeHz...	[null]	false	python6			python@gmail.cc

Change_password Api :

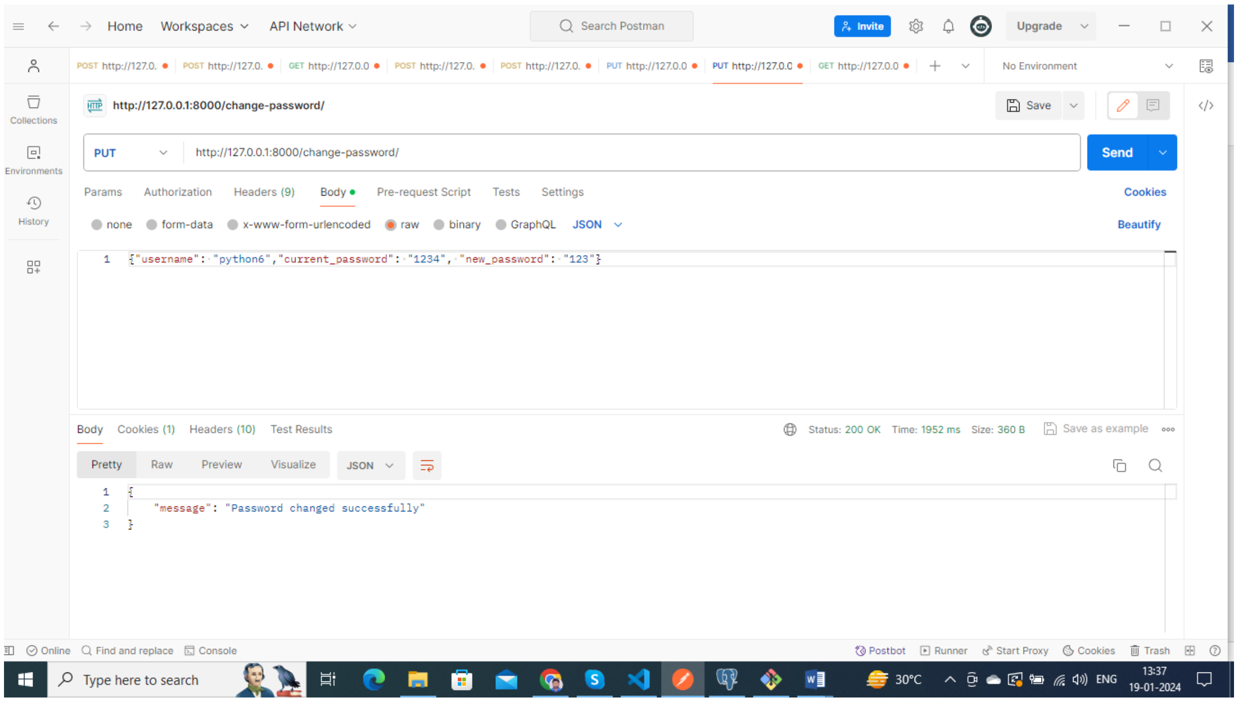
Similar to Reset Api, it will check whether user exsist , if that user exsist then it check old password is valid in database or not , if that old password is valid then it will update that old password with new password

```
def change_password(request):
    if request.method == "PUT":
        current_password = request.data['current_password']
        new_password = request.data['new_password']
        username = request.data['username']
        if not all([current_password, new_password,username]):
            return Response({'message': "Both current and new
passwords are required."}, status=status.HTTP_400_BAD_REQUEST)
        try:
            user_details =
User.objects.get(username=username)
            if
check_password(current_password,user_details.password):
                #user, created =
User.objects.update_or_create(username=username,
defaults={'password': new_password})                  # this
method will automatically replace and save paticular feild , but
it won't encript password
                user=User.objects.get(username=username)
                user.set_password(new_password)
                       #set_password is inbulit method which
will encrpt new password
                user.save()
                       #it will save encrypted password
                return Response({'message': "Password changed
successfully"})
            return Response({"message": "current password is
incorrect"})
```

```
        except:
            return Response({'message': "User with that details
not found."}, status=status.HTTP_400_BAD_REQUEST)
    return Response(status=status.HTTP_200_OK)
```

Performing Unit Testing :

With the help of Postman tool, we will be performing various test according to input



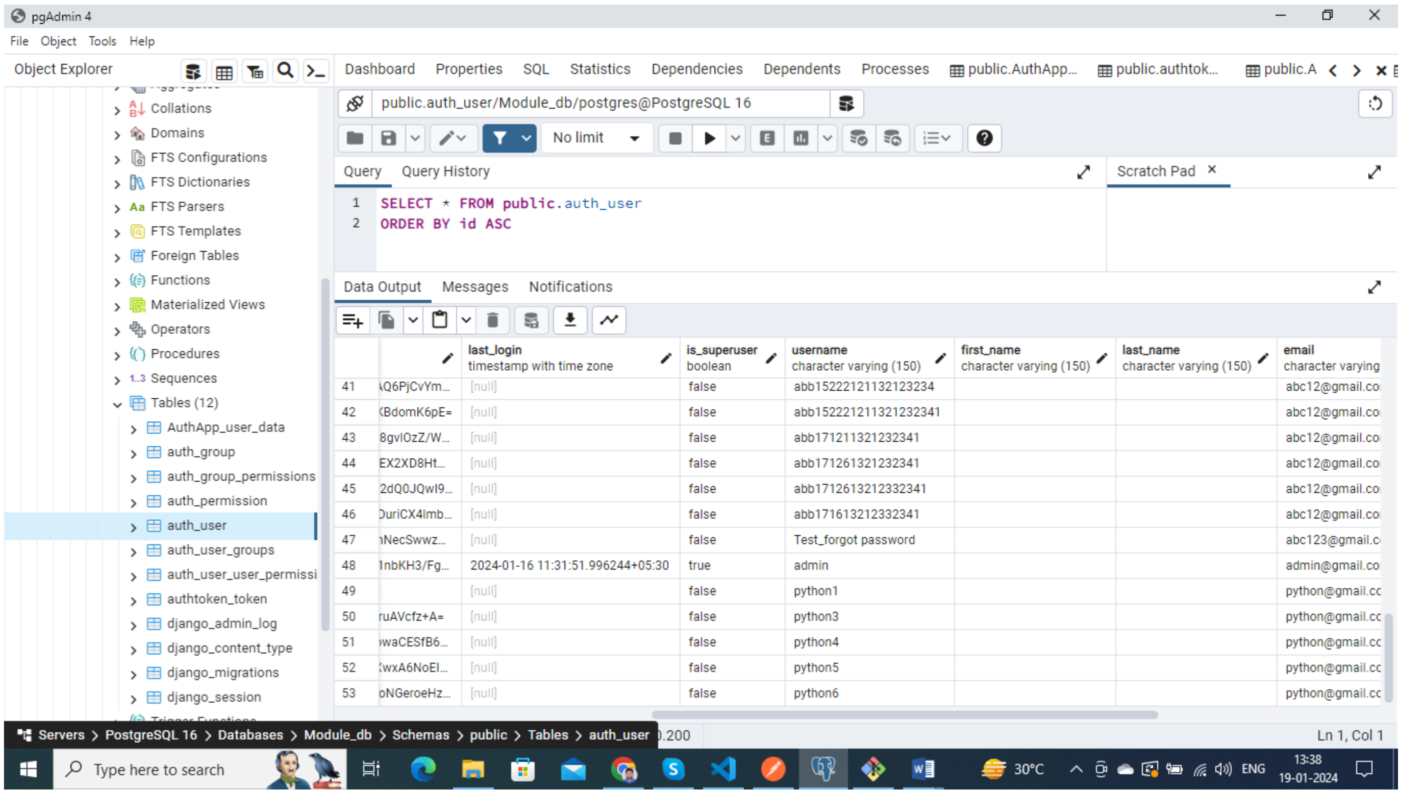
http://127.0.0.1:8000/change-password/

PUT http://127.0.0.1:8000/change-password/

```
{"username": "python6","current_password": "1234", "new_password": "123"}
```

Body Cookies (1) Headers (10) Test Results — Status: 200 OK Time: 1952 ms Size: 360 B

```
{
    "message": "Password changed successfully"
}
```



public.auth_user/Module_db/postgres@PostgreSQL 16

```
SELECT * FROM public.auth_user
ORDER BY id ASC
```

		last_login timestamp with time zone	is_superuser boolean	username character varying (150)	first_name character varying (150)	last_name character varying (150)	email character varying
41	\Q6PjCvYm...	[null]	false	abb15222121132123234			abc12@gmail.co
42	(BdomK6pE=	[null]	false	abb152221211321232341			abc12@gmail.co
43	8gvlOzZ/W...	[null]	false	abb171211321232341			abc12@gmail.co
44	EX2XD8Ht...	[null]	false	abb171261321232341			abc12@gmail.co
45	2dQ0JQwl9...	[null]	false	abb1712613212332341			abc12@gmail.co
46	DuriCX4lmb...	[null]	false	abb171613212332341			abc12@gmail.co
47	ıNecSwwz...	[null]	false	Test_forgot password			abc123@gmail.c
48	1nbKH3/Fg...	2024-01-16 11:31:51.996244+05:30	true	admin			admin@gmail.co
49		[null]	false	python1			python@gmail.cc
50	ruAVcfz+A=	[null]	false	python3			python@gmail.cc
51	›waCESfB6...	[null]	false	python4			python@gmail.cc
52	(wxA6NoEl...	[null]	false	python5			python@gmail.cc
53	oNGeroeHz...	[null]	false	python6			python@gmail.cc

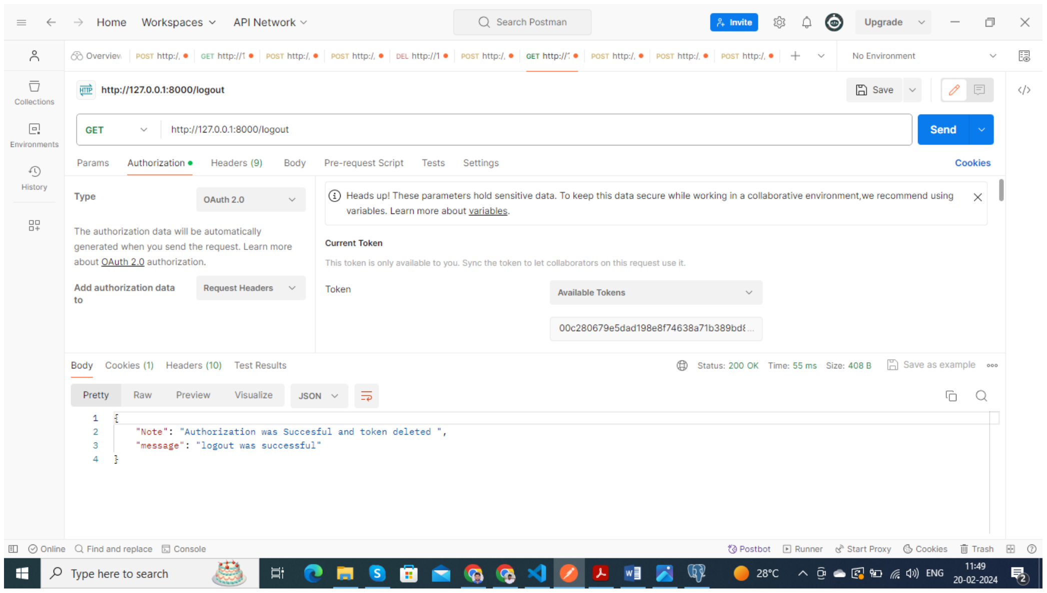
Logout Api:

According to the concept of Authentication and reference of jwt token, for login we will be having token , if we delete that token then that user will be automatically logged out

```
def logout(request):
    request.user.auth_token.delete()
    return Response({"Note" :
"Authorization was Succesful and token
deleted ",
        "message": "logout was
successful"})
```

Performing Unit Testing :

With the help of Postman tool, we will be performing various test according to input



GET http://127.0.0.1:8000/logout

Type: OAuth 2.0

Add authorization data to: Request Headers

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.

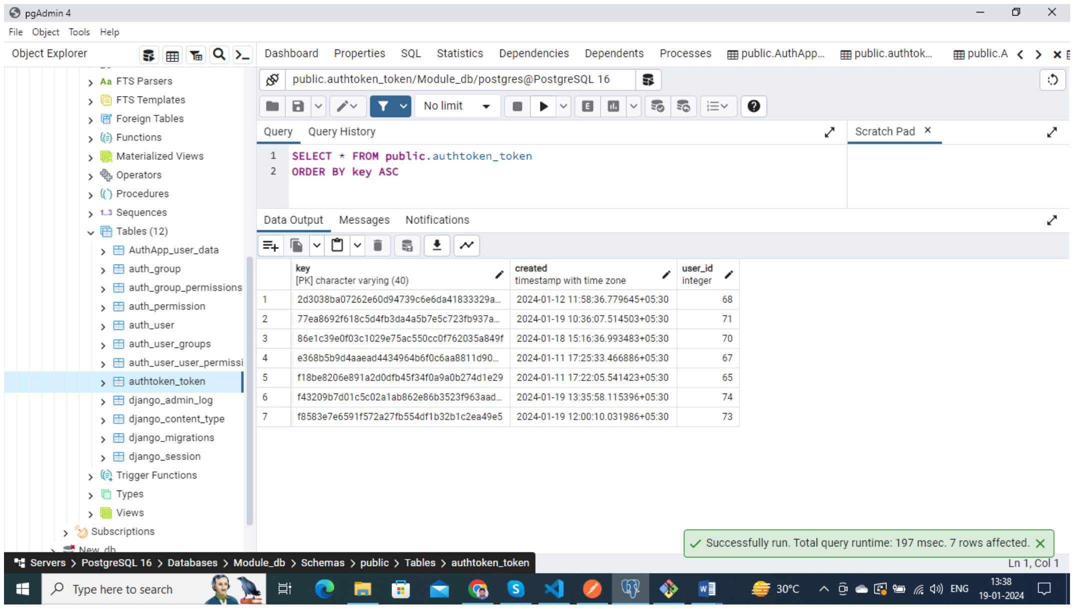
Current Token

This token is only available to you. Sync the token to let collaborators on this request use it.

Token: 00c280679e5dad198e8f74638a71b389bd…

Status: 200 OK Time: 55 ms Size: 408 B

```
{
    "Note": "Authorization was Succesful and token deleted ",
    "message": "logout was successful"
}
```



public.authtoken_token/Module_db/postgres@PostgreSQL 16

```
SELECT * FROM public.authtoken_token
ORDER BY key ASC
```

	key [PK] character varying (40)	created timestamp with time zone	user_id integer
1	2d3038ba07262e60d94739c6e6da41833329a…	2024-01-12 11:58:36.779645+05:30	68
2	77ea8692f618c5d4fb3da4a5b7e5c723fb937a…	2024-01-19 10:36:07.514503+05:30	71
3	86e1c39e0f03c1029e75ac550cc0f762035a849f	2024-01-18 15:16:36.993483+05:30	70
4	e368b5b9d4aaead4434964b6f0c6aa8811d90…	2024-01-11 17:25:33.466886+05:30	67
5	f18be8206e891a2d0dfb45f34f0a9a0b274d1e29	2024-01-11 17:22:05.541423+05:30	65
6	f43209b7d01c5c02a1ab862e86b3523f963aad…	2024-01-19 13:35:58.115396+05:30	74
7	f8583e7e6591f572a27fb554df1b32b1c2ea49e5	2024-01-19 12:00:10.031986+05:30	73

Successfully run. Total query runtime: 197 msec. 7 rows affected.

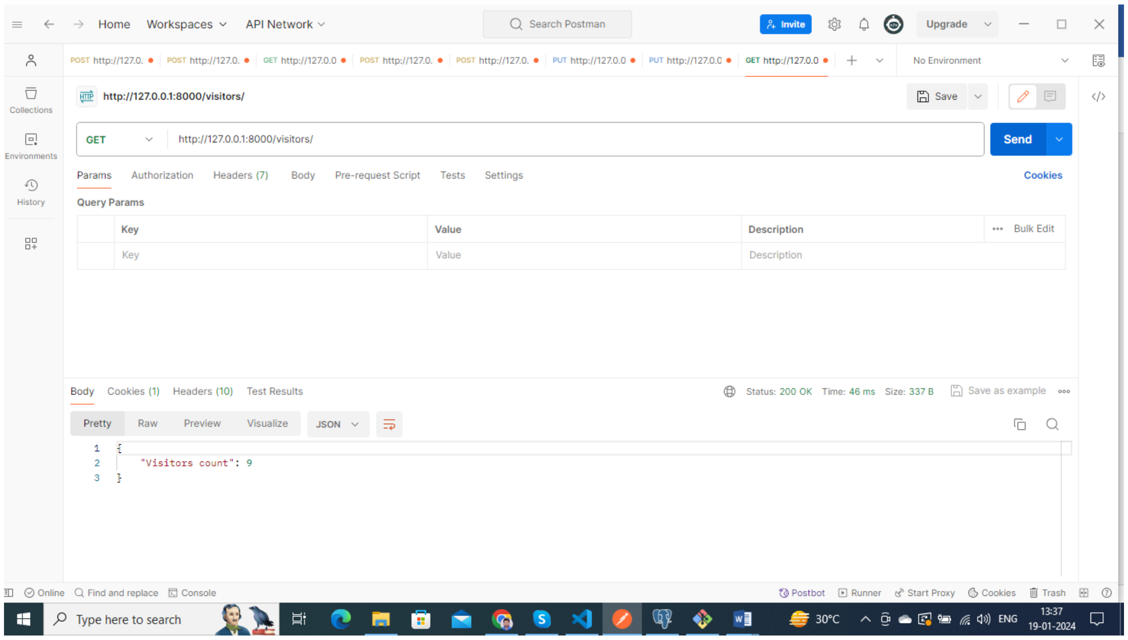
Visitors Api:

Here, in this Api, we will track no of visitors for our website so, that when ever a user is logged in we will automatically increment count attribute value in our database , so that we can be able to get no of visitors to our page till date and we can also get how many times a paticular person visited

```
def visitors(request):
    visitors_count=0
    data=User_data.objects.all()
    serilizer=user_data_seralizer(data,many=True)
    for value in serilizer.data:
        visitors_count+=value['count']
    return Response({"Visitors count":visitors_count})
```

Performing Unit Testing :

With the help of Postman tool, we will be performing various test according to input



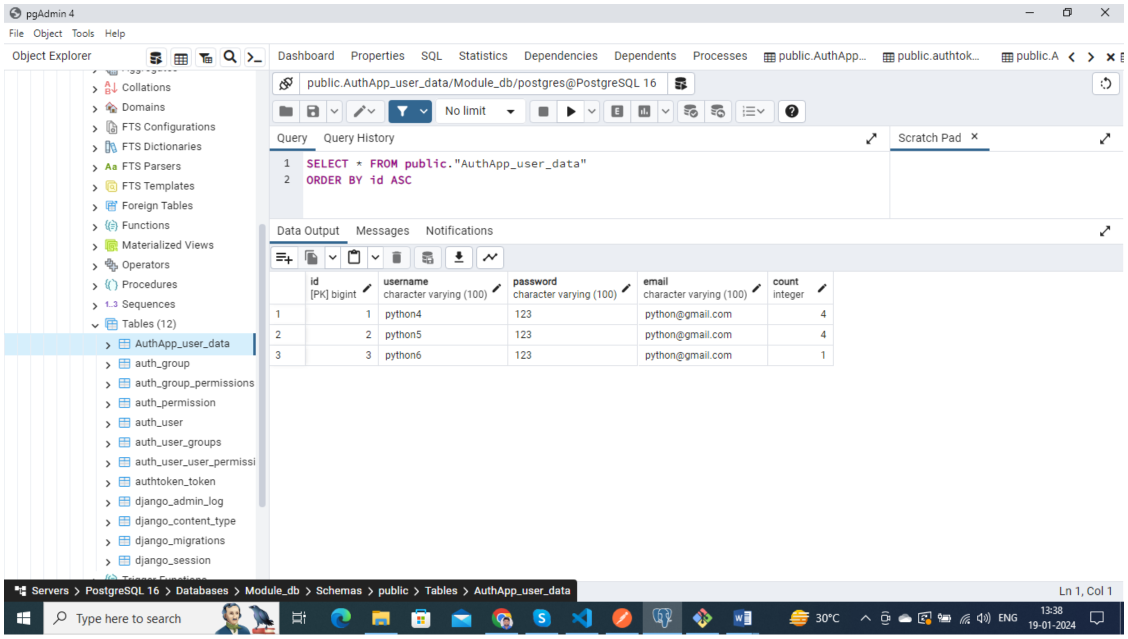
GET http://127.0.0.1:8000/visitors/

Key	Value	Description
Key	Value	Description

Body | Cookies (1) | Headers (10) | Test Results

Status: 200 OK Time: 46 ms Size: 337 B

```
{
    "Visitors count": 9
}
```



public.AuthApp_user_data/Module_db/postgres@PostgreSQL 16

```
SELECT * FROM public."AuthApp_user_data"
ORDER BY id ASC
```

	id [PK] bigint	username character varying (100)	password character varying (100)	email character varying (100)	count integer
1	1	python4	123	python@gmail.com	4
2	2	python5	123	python@gmail.com	4
3	3	python6	123	python@gmail.com	1

THANK YOU