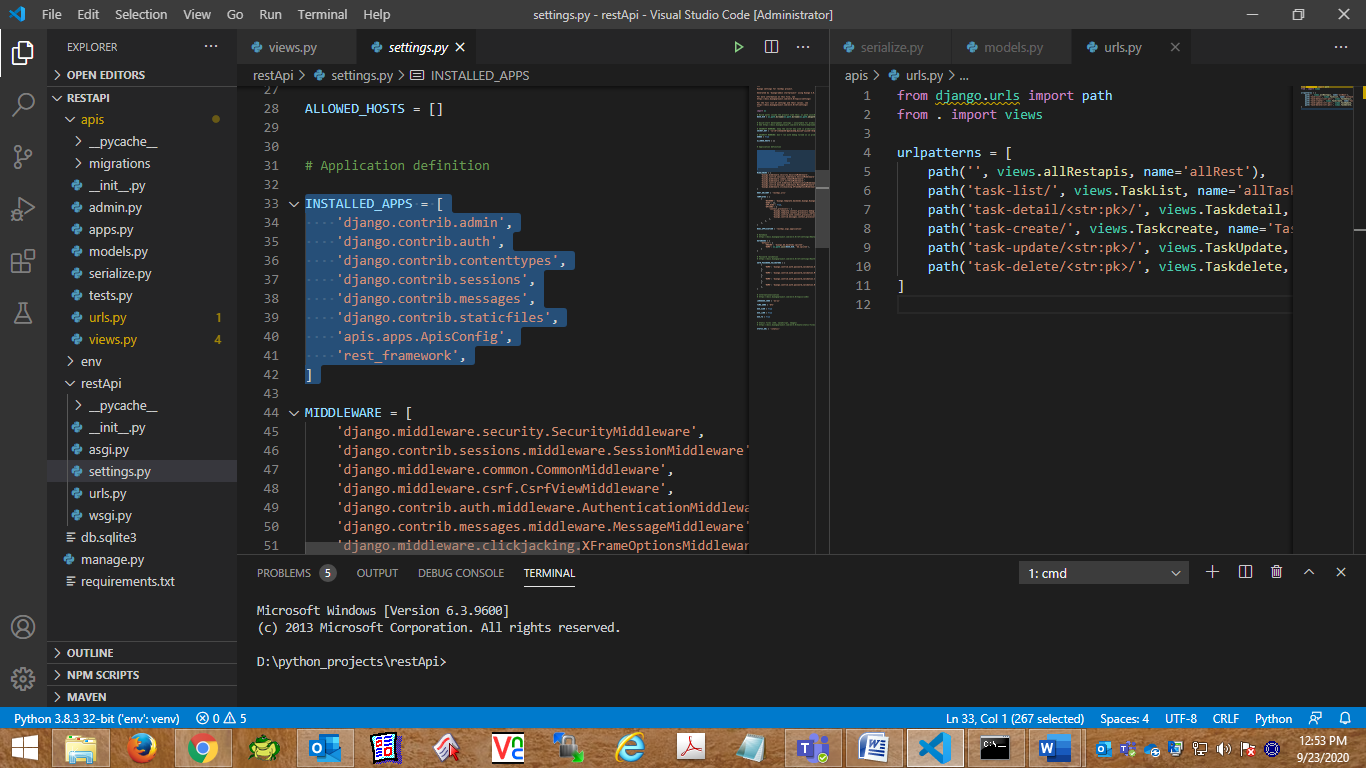
pip install djangorestframework

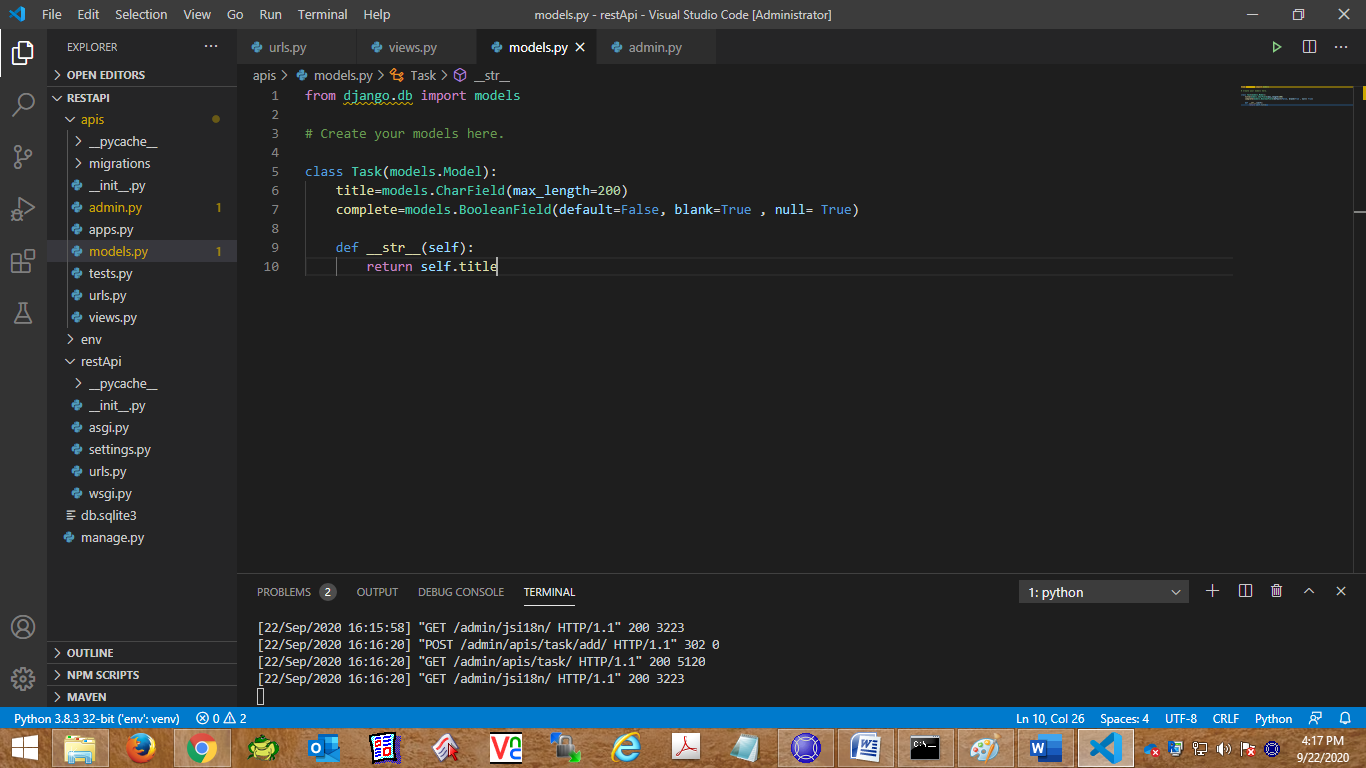
We had created a project and app then installed in locally virtual environment.

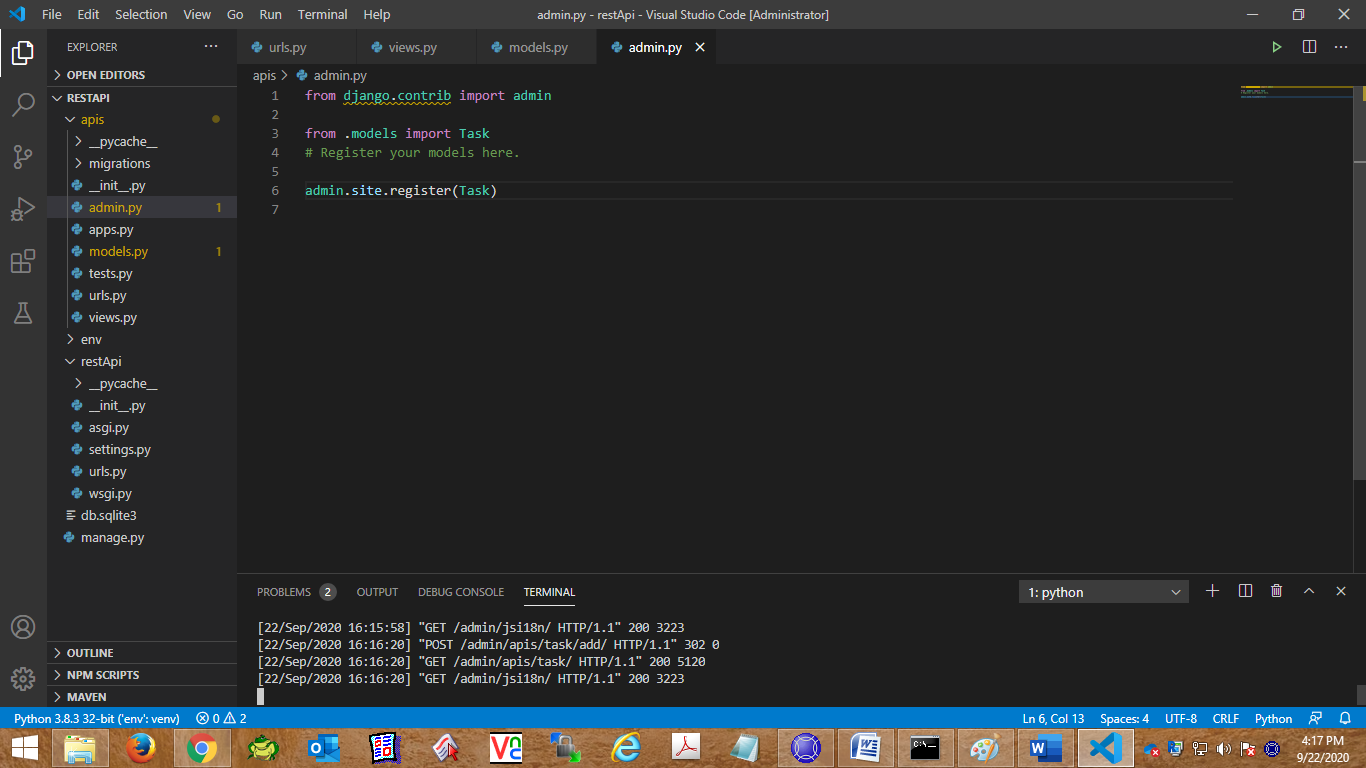
Add…

'rest\_framework',



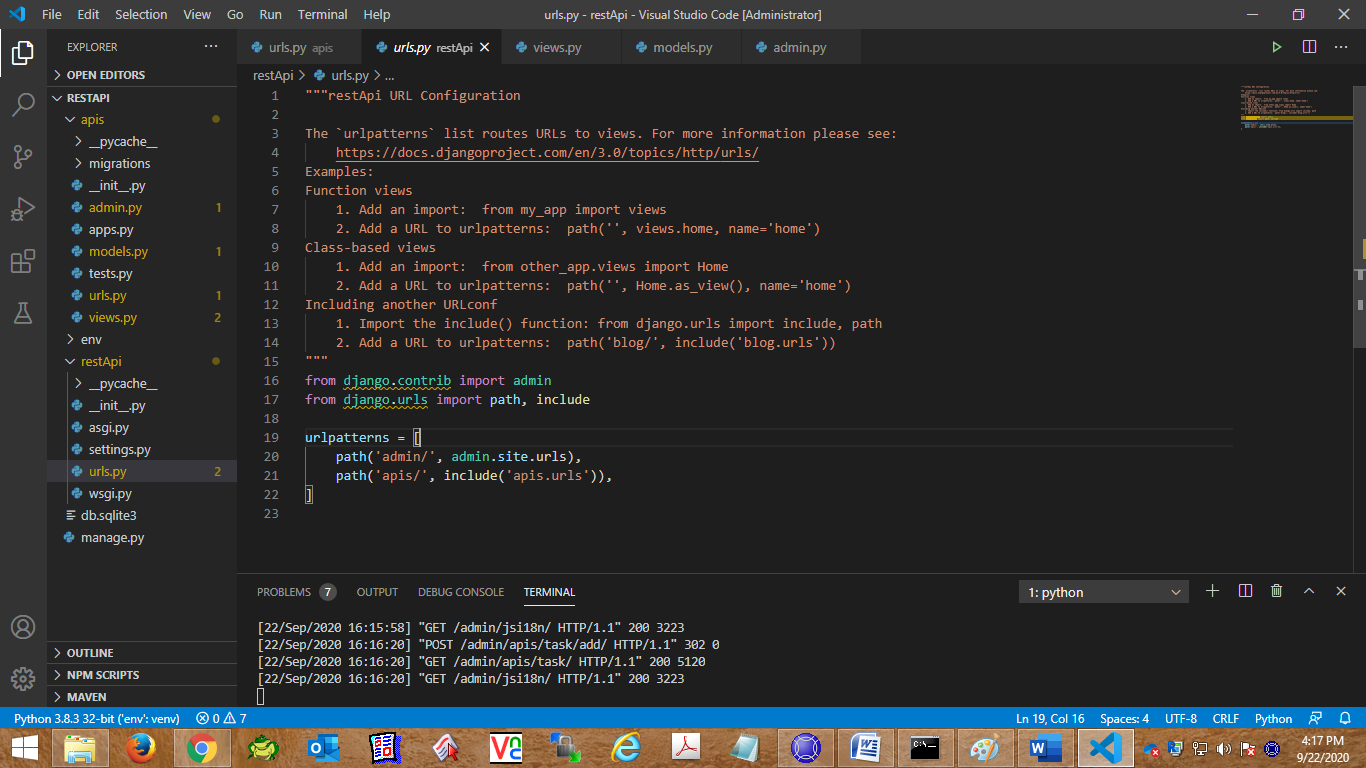
Now let’s create a Model class named as Task. Add some values using admin.



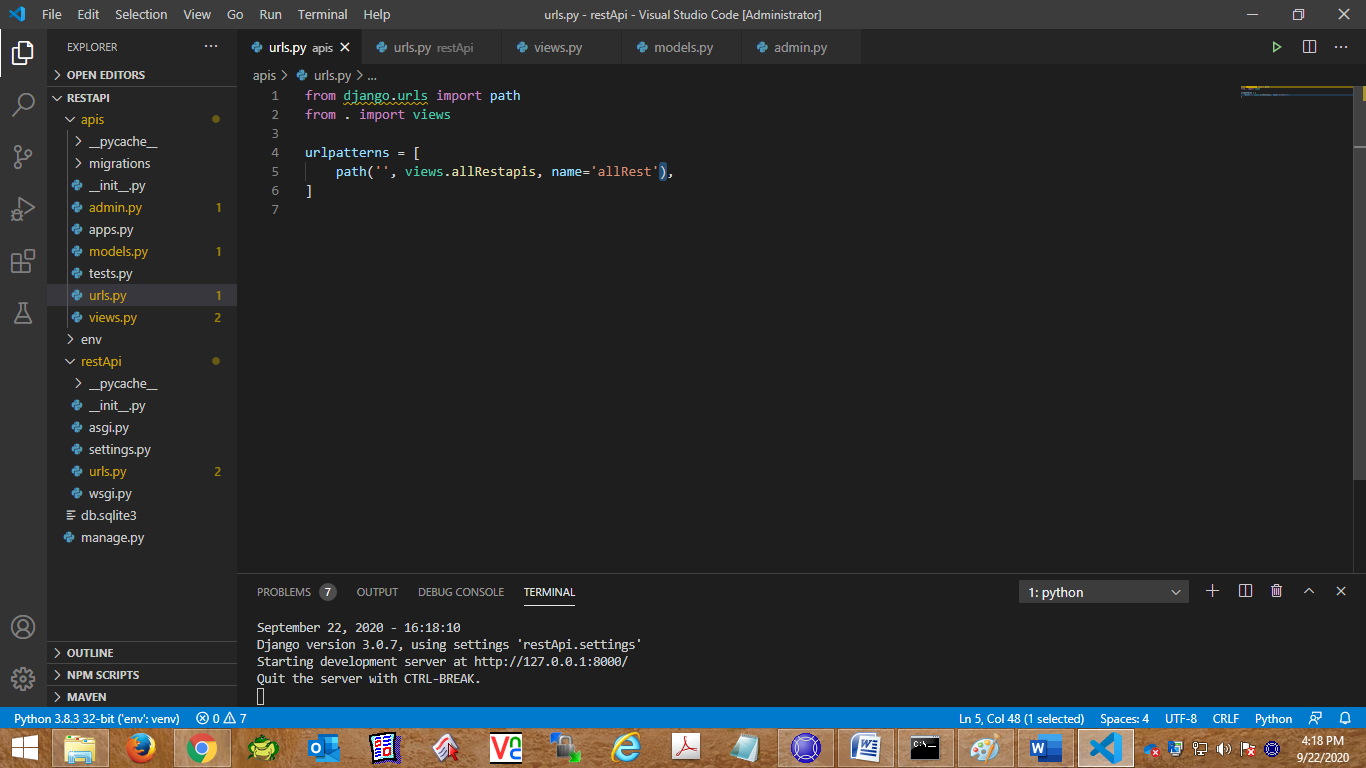


Add Urls configuration.

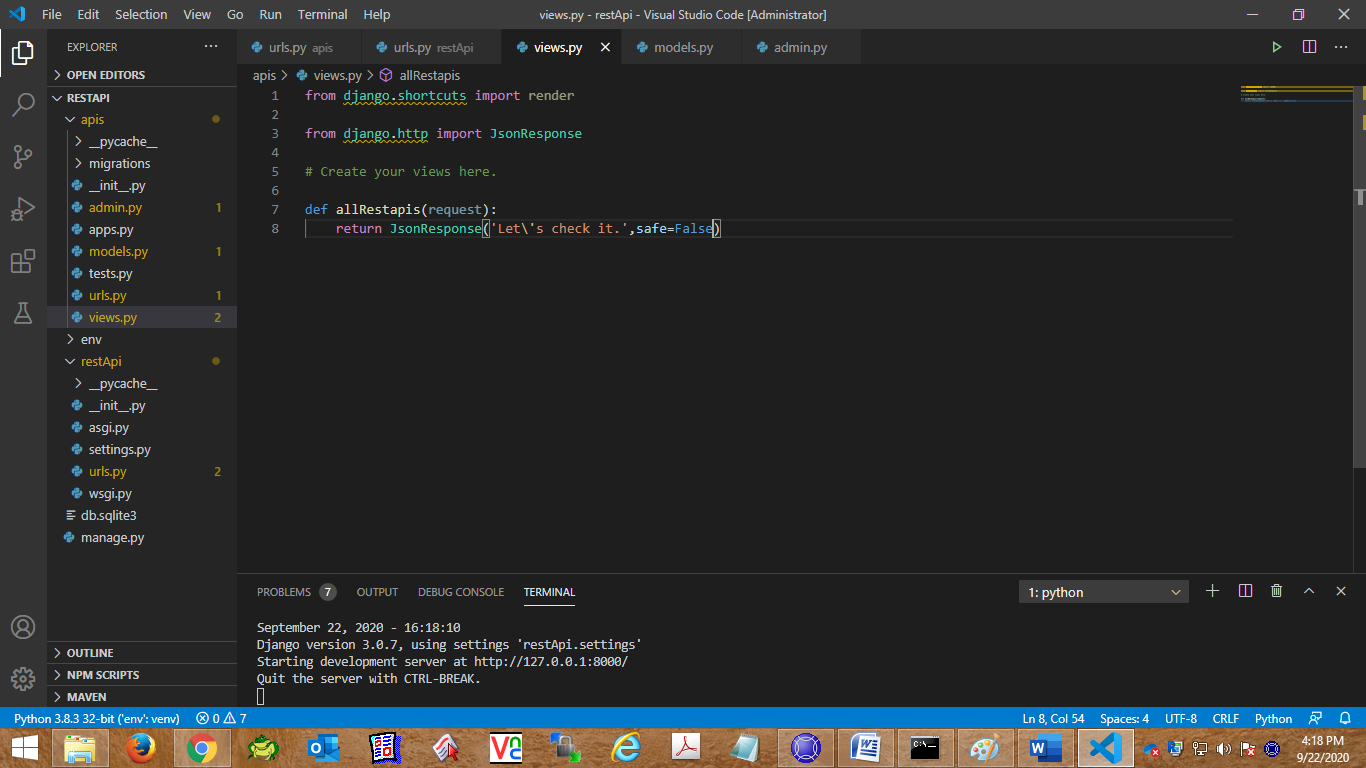
Main Url file:

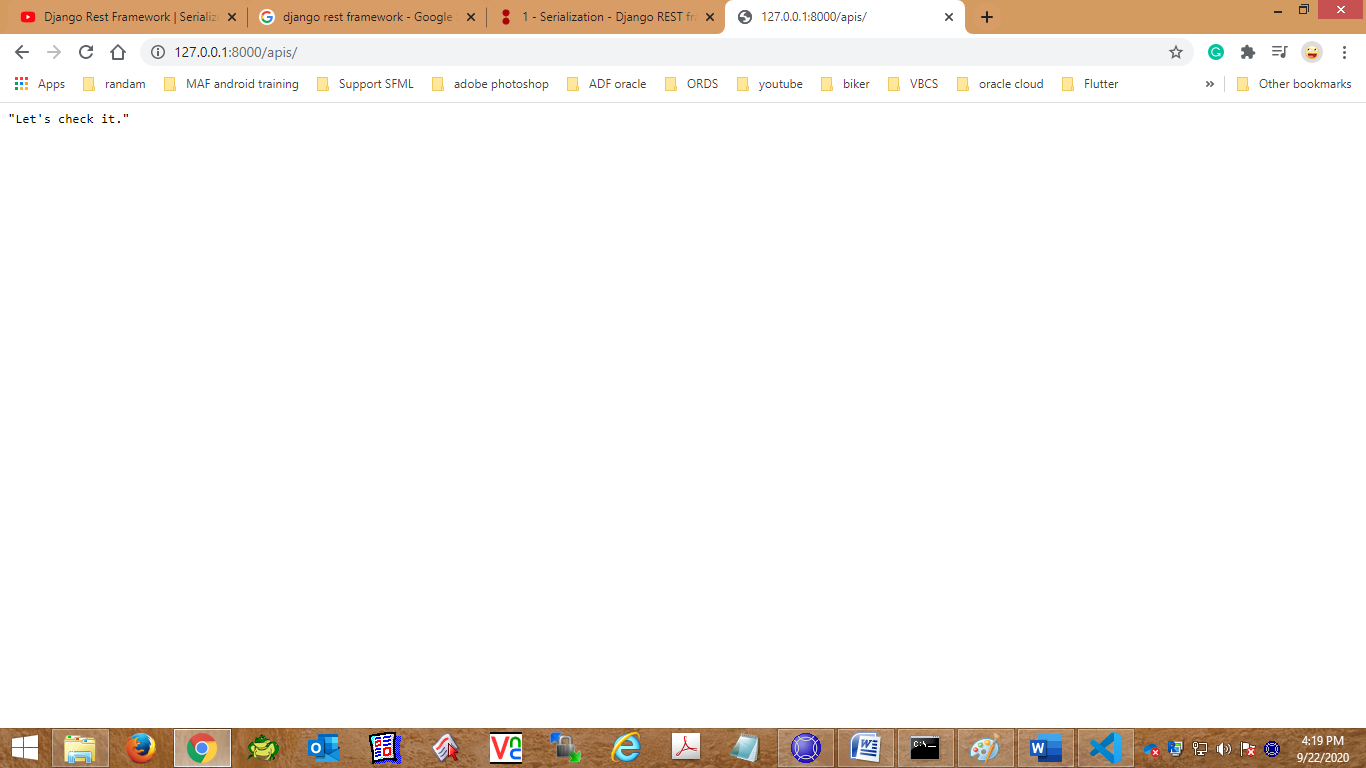


App url file.



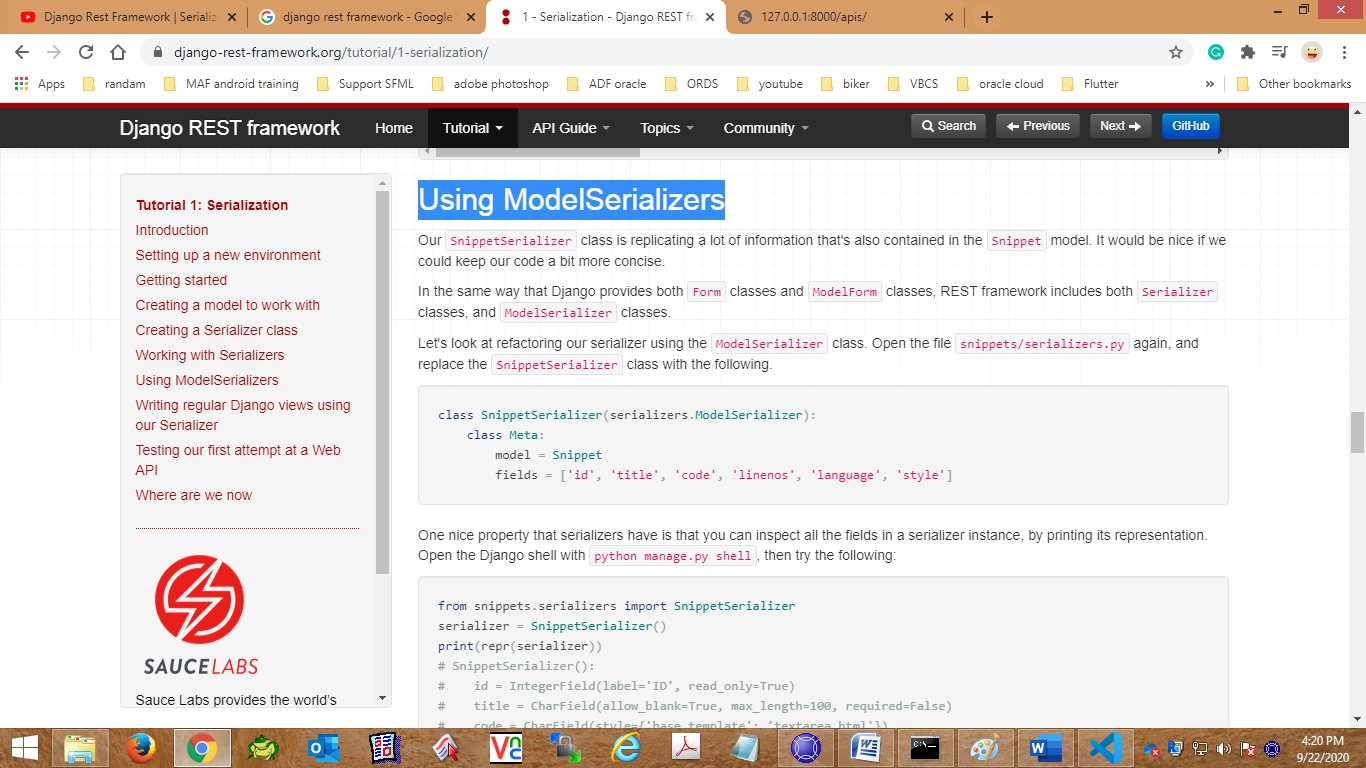
Just for testing return JsonResponse





Response is coming, Now start our rest Frame Work working.

We are [Using ModelSerializers](https://www.django-rest-framework.org/tutorial/1-serialization/#using-modelserializers)



Create a **serialize.py** file in apps level and mention serializer Model Class in it.

from rest\_framework import serializers

from rest\_framework import serializers

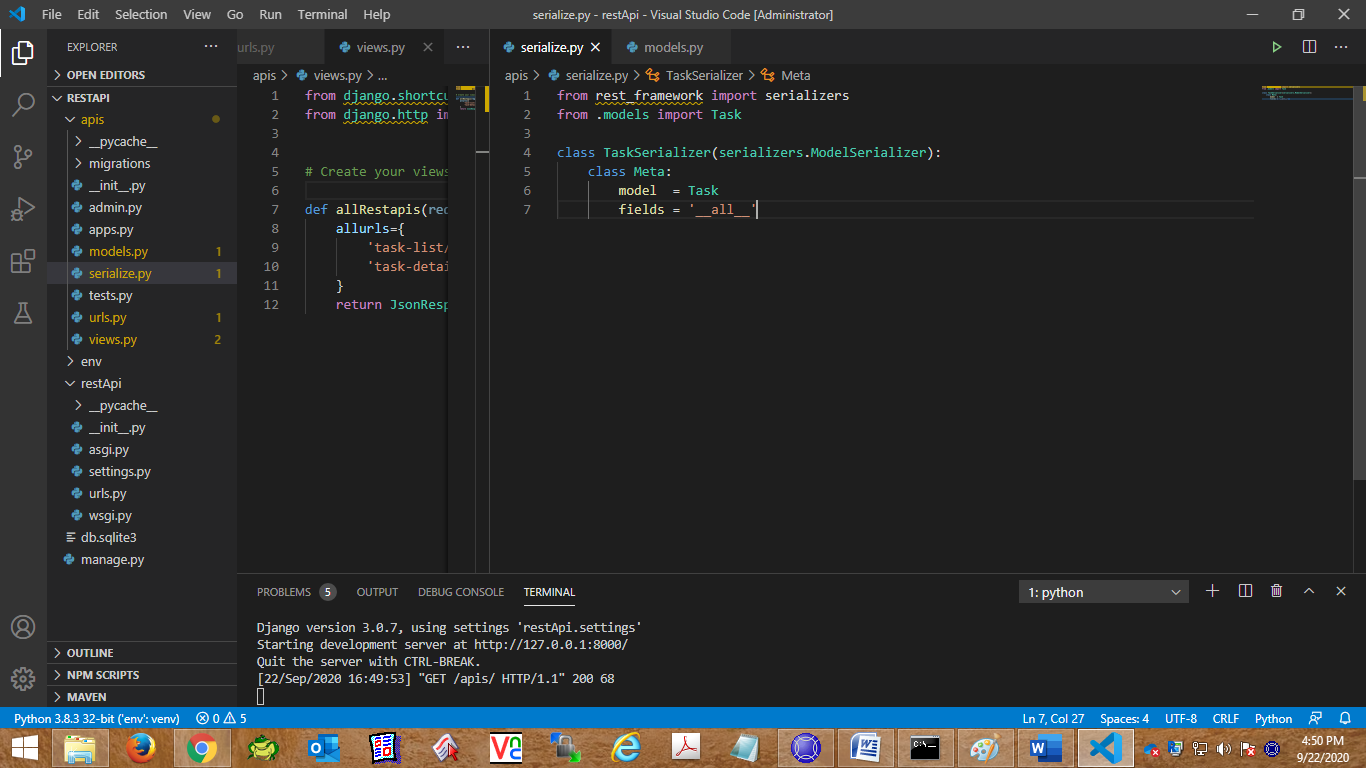
from .models import Task

class TaskSerializer(serializers.ModelSerializer):

    class Meta:

        model  = Task

        fields = '\_\_all\_\_'



Now we are going to modify views.py file to response and request for **Rest Frame work API call**

[**Tutorial 2: Requests and Responses**](https://www.django-rest-framework.org/tutorial/2-requests-and-responses/#tutorial-2-requests-and-responses)

[**Wrapping API views**](https://www.django-rest-framework.org/tutorial/2-requests-and-responses/#wrapping-api-views)**:**

1. The @api\_view decorator for working with function based views.

@api\_view(['GET'])

from rest\_framework.decorators import api\_view

from rest\_framework.response import Response

from django.shortcuts import render

from django.http import JsonResponse

from rest\_framework.decorators import api\_view

from rest\_framework.response import Response

# Create your views here.

# For General list we are creating this

@api\_view(['GET'])

def allRestapis(request):

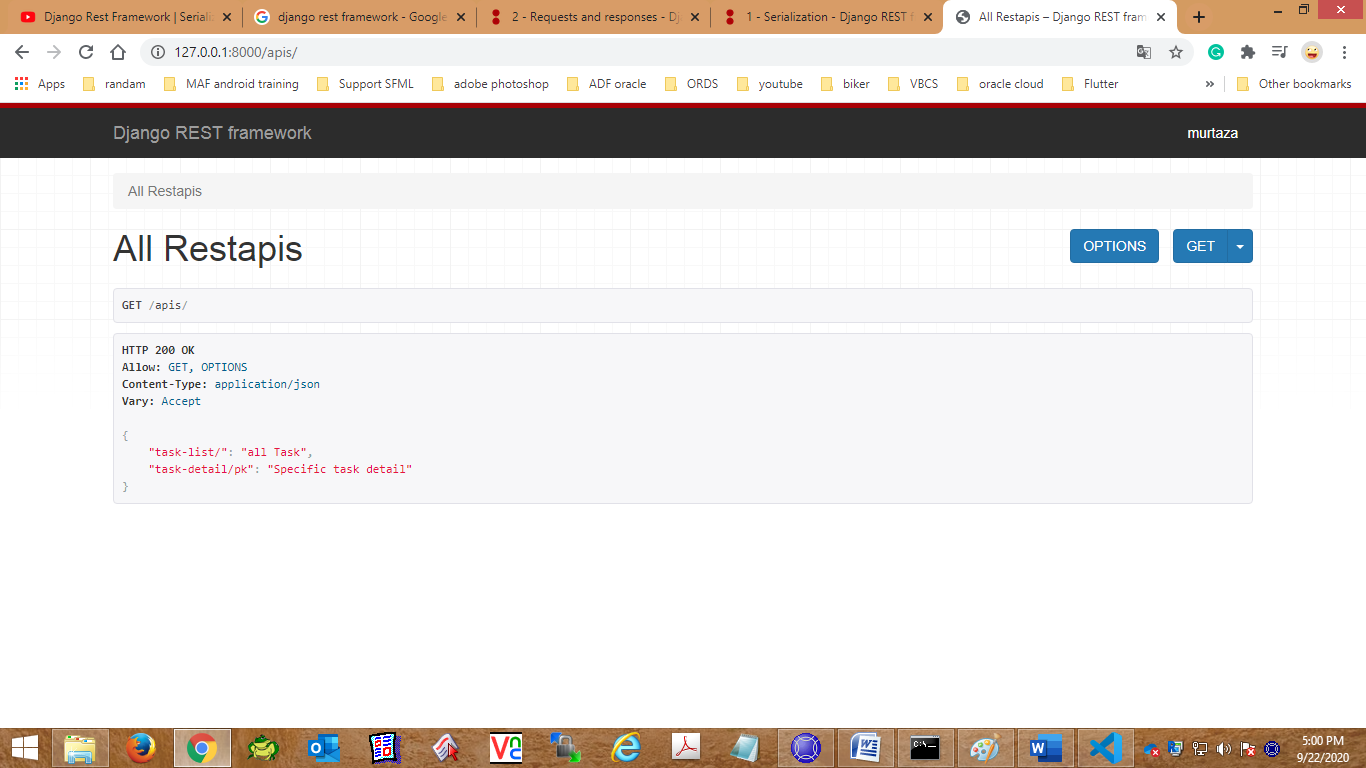
    allurls={

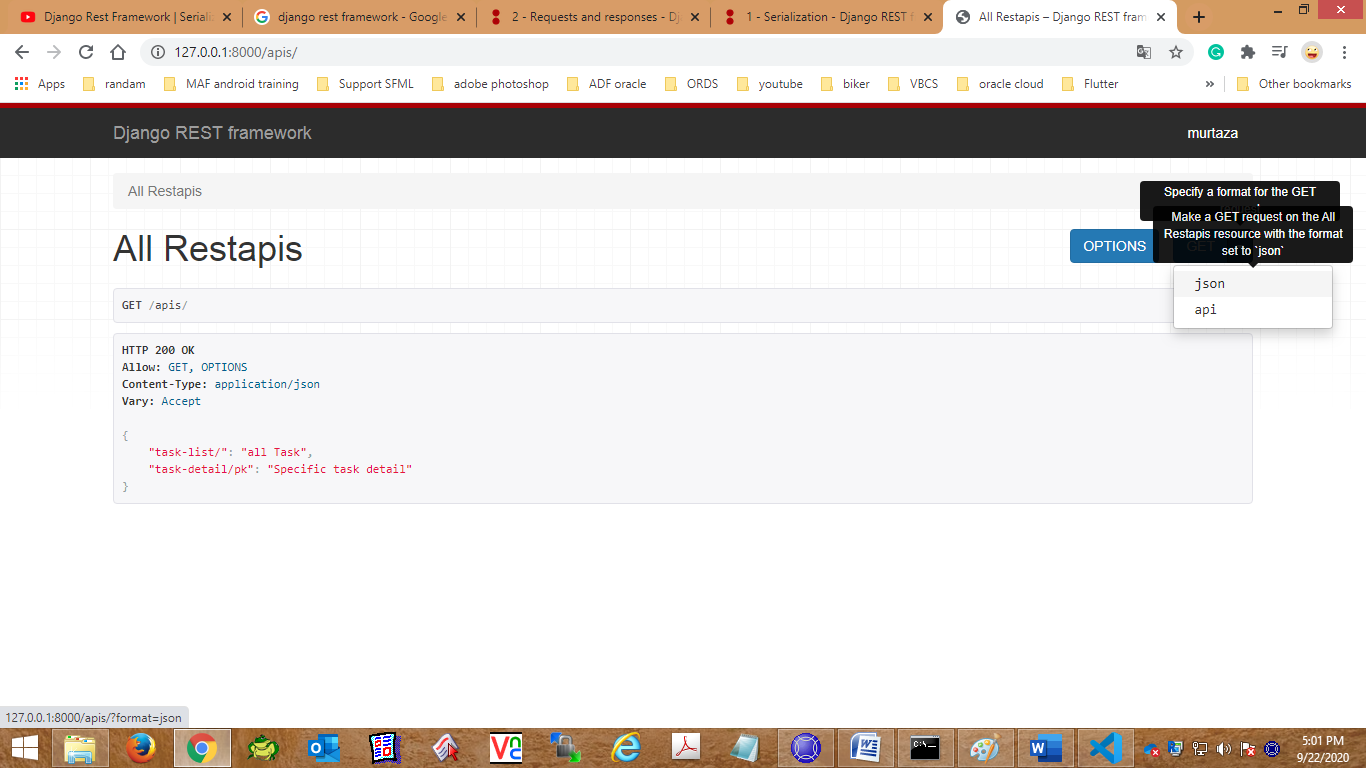
        'task-list/':'all Task',

        'task-detail/pk':'Specific task detail'

    }

    return Response(allurls)





In case you select **json.**

{"task-list/":"all Task","task-detail/pk":"Specific task detail"}

For all Task list. Function will be like that.

from .models import Task

from .serialize import TaskSerializer

and function be like:

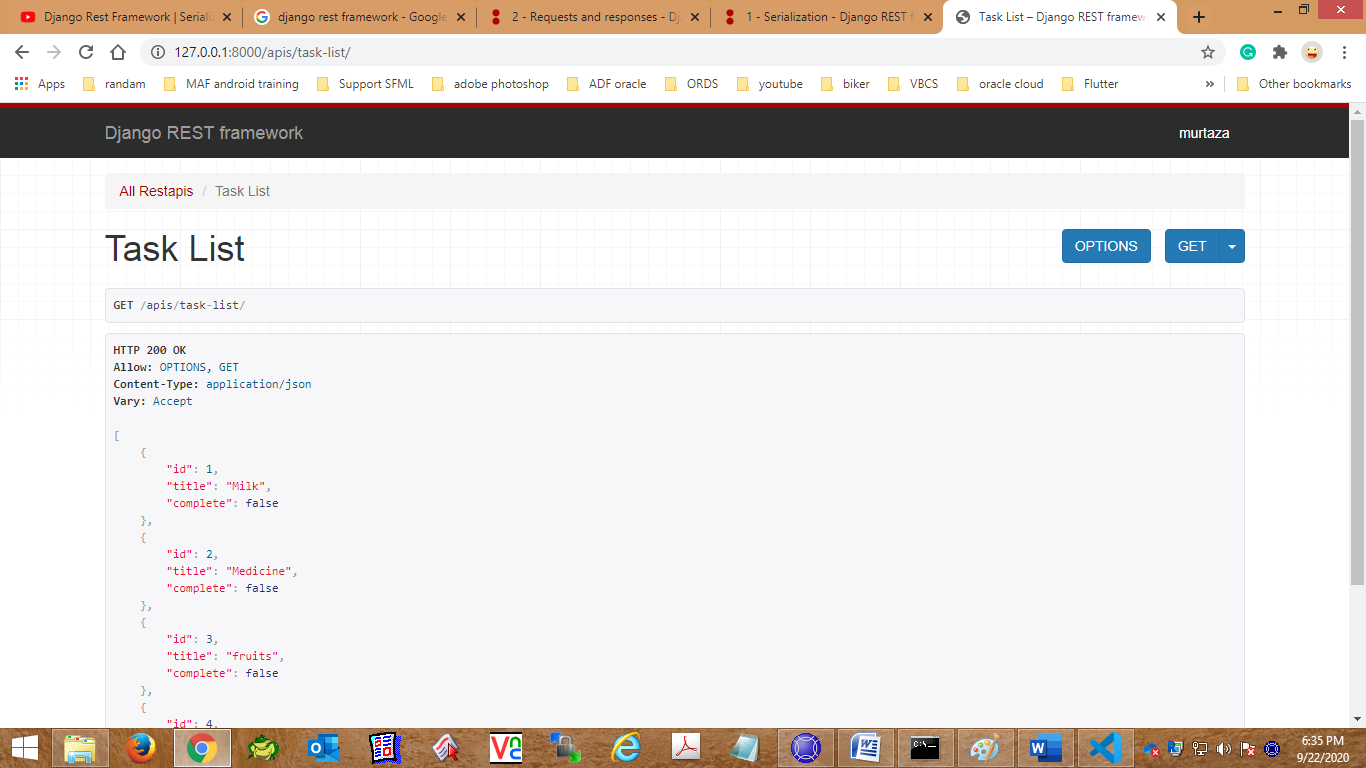
@api\_view(['GET'])

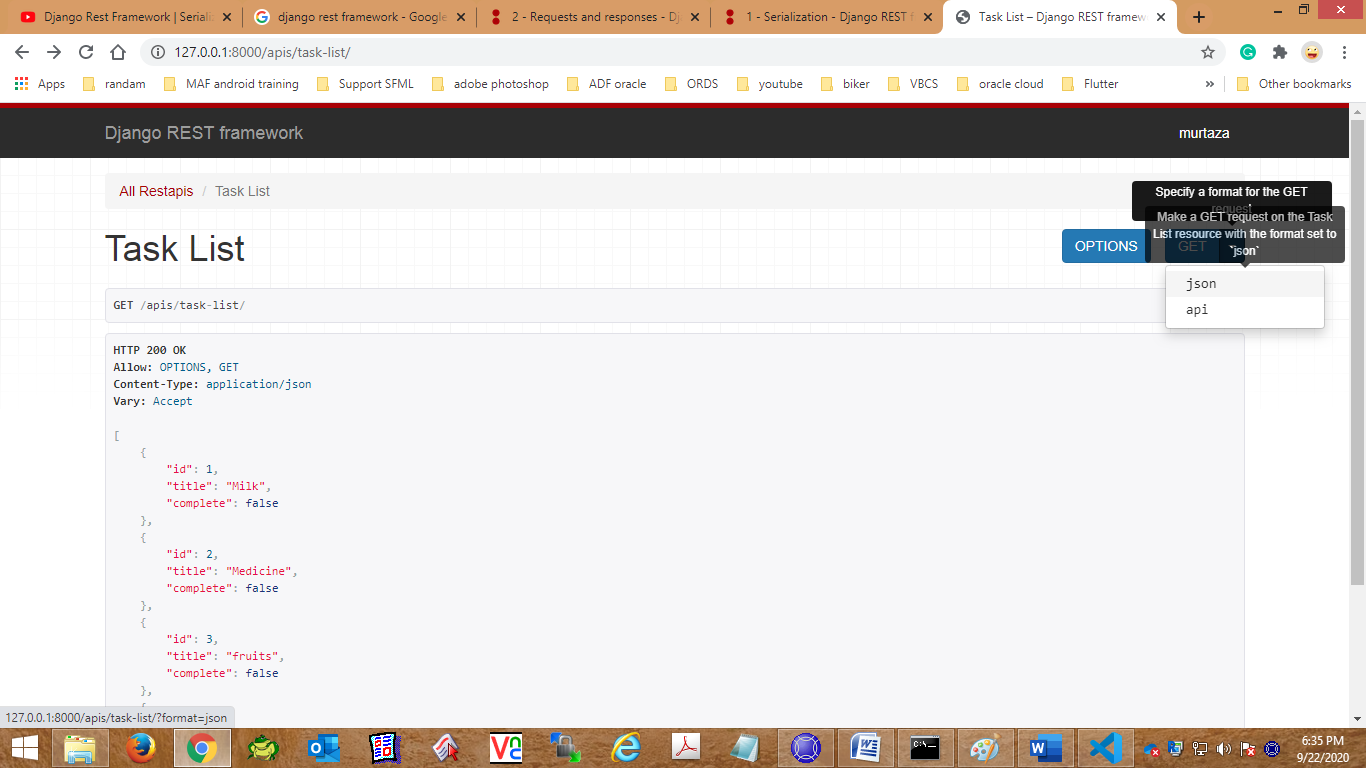
def TaskList(request):

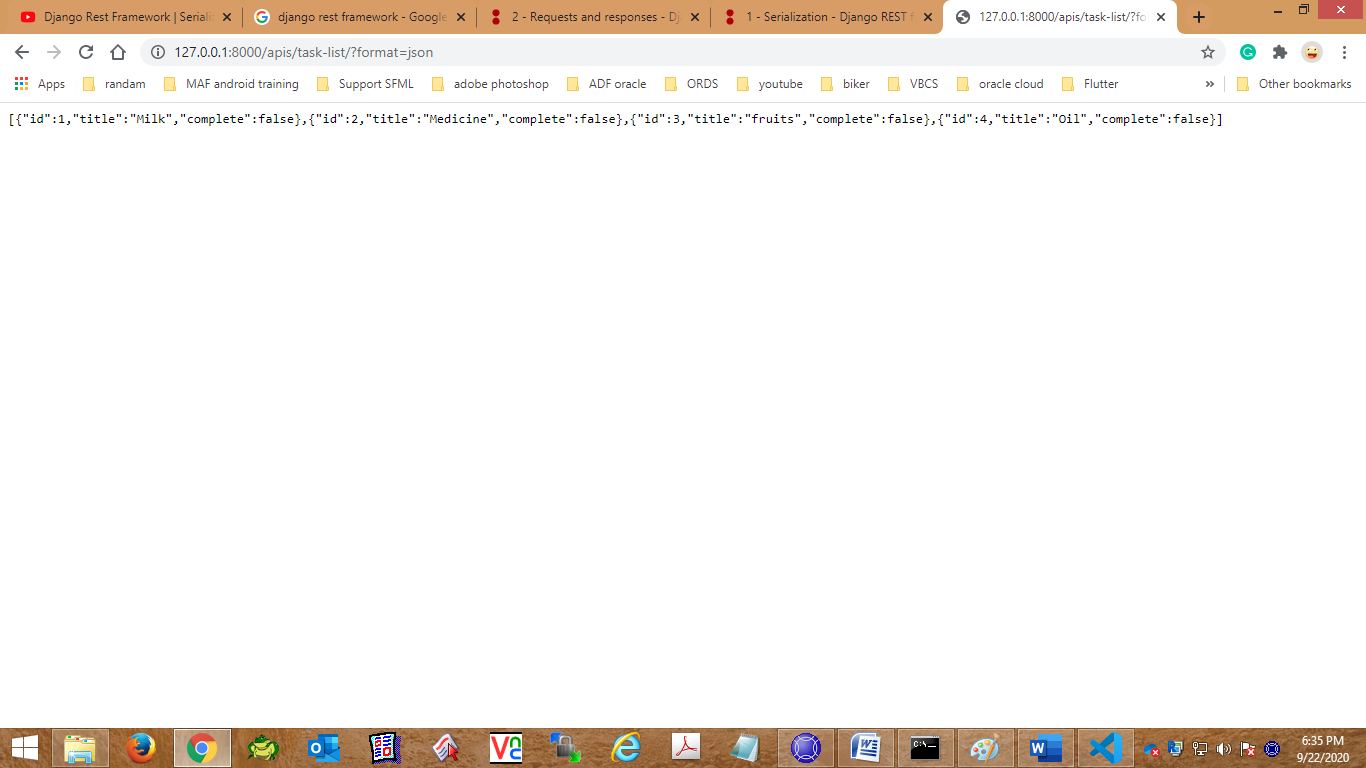
    task=Task.objects.all()

    serialize=TaskSerializer(task,many=True)

    return Response(serialize.data)







To get only specific item.

@api\_view(['GET'])

def Taskdetail(request, pk):

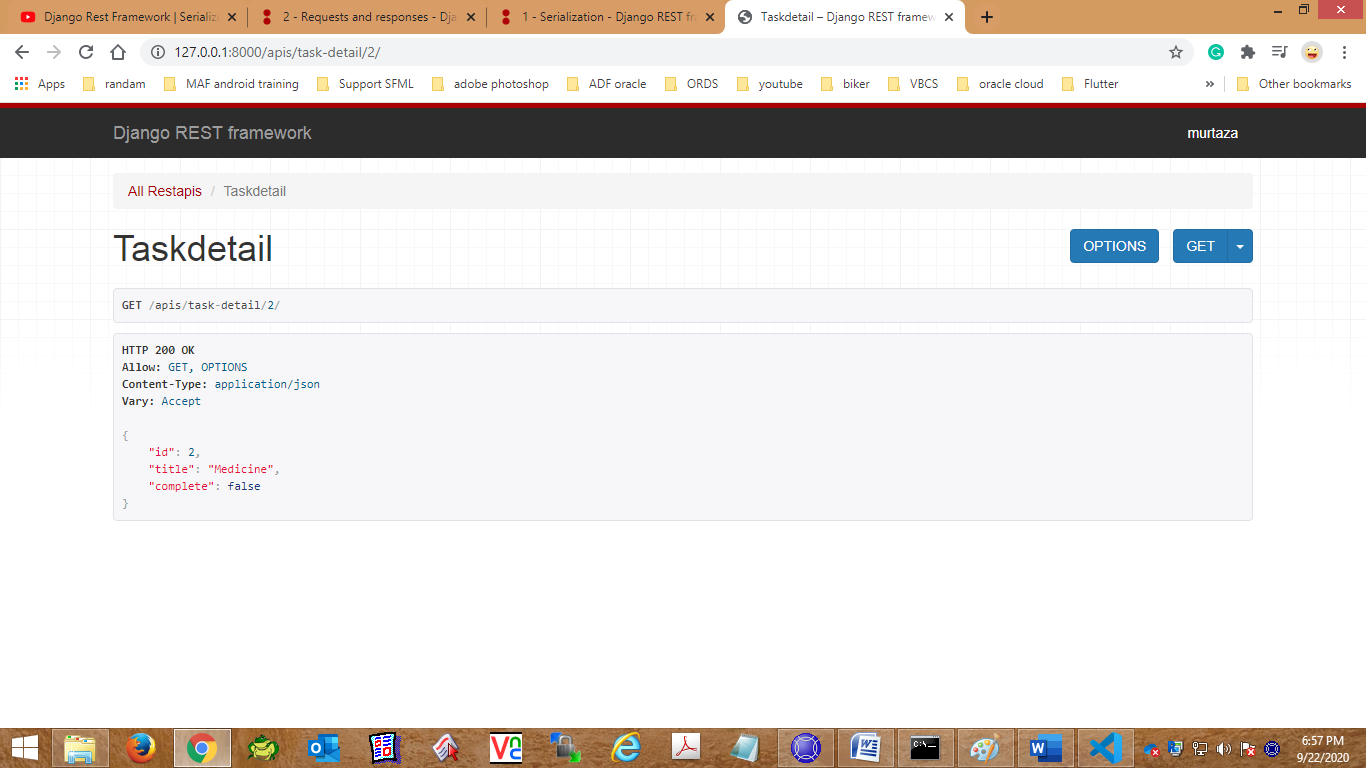
    task=Task.objects.get(id=pk)

    serialize=TaskSerializer(task,many=False)

    return Response(serialize.data)

URL be like:

path('task-detail/<str:pk>/', views.Taskdetail, name='Task-detail'),



For create a task.

url be like:

path('task-create/', views.Taskcreate, name='Task-create'),

and function be like:

@api\_view(['POST'])

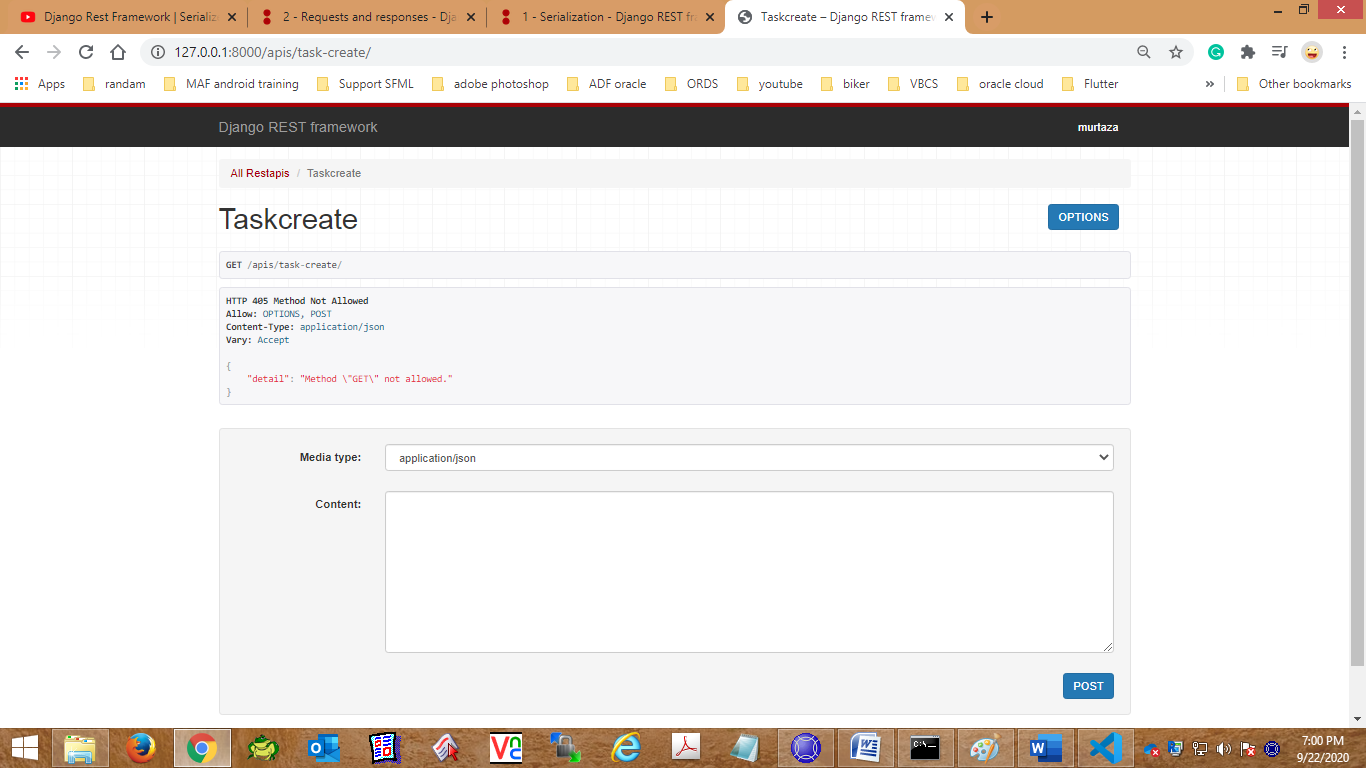
def Taskcreate(request):

    serialize=TaskSerializer(data=request.data)

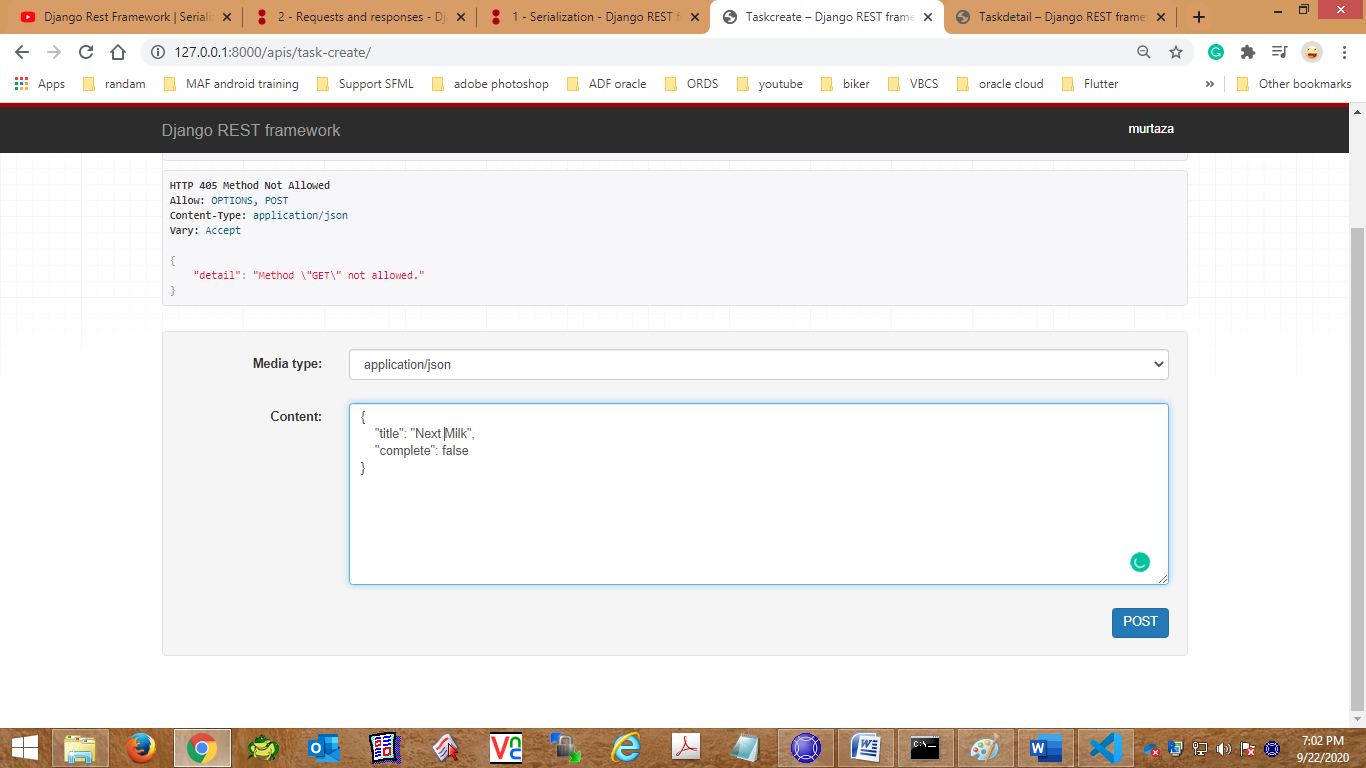
    if serialize.is\_valid():

        serialize.save()

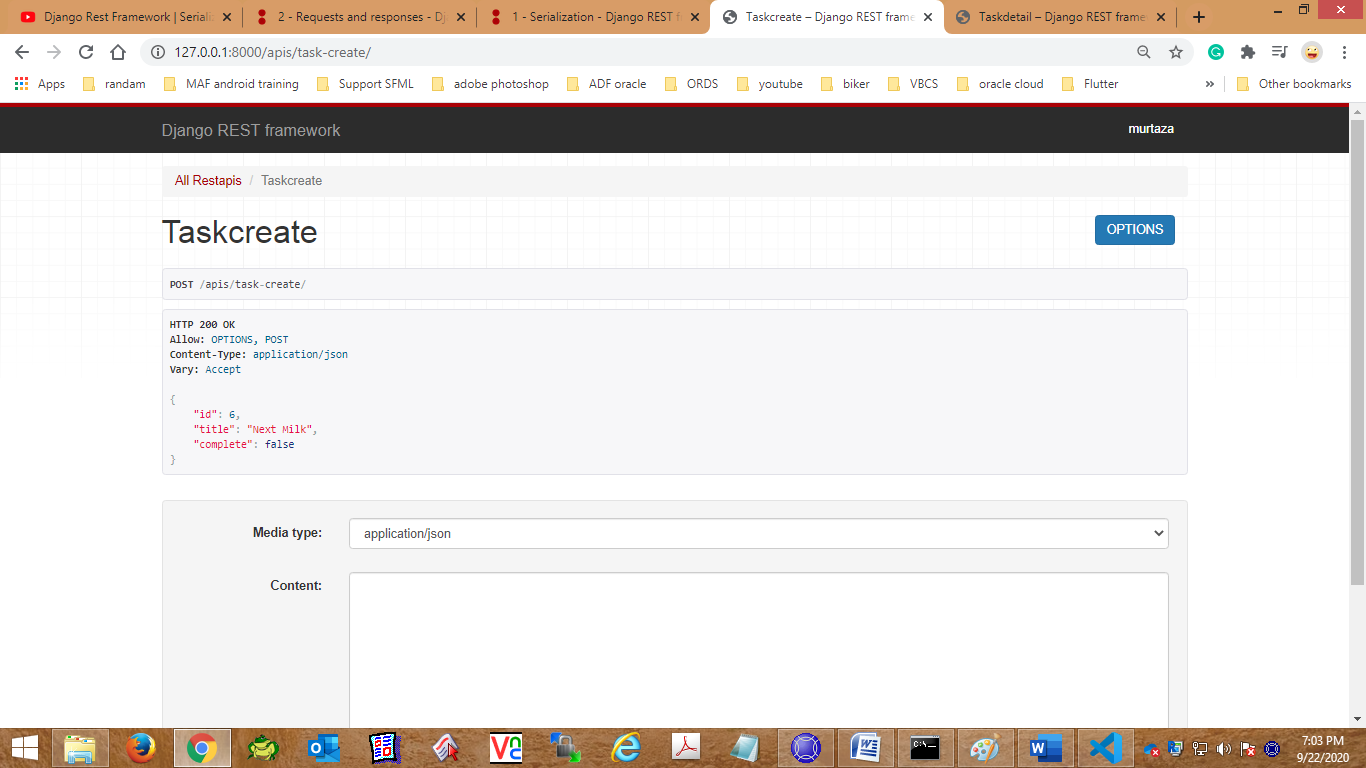
    return Response(serialize.data)



Payload be like:



And response will return as per mention in function.



# For update specific:

Url be like :

path('task-update/<str:pk>/', views.TaskUpdate, name='Task-update'),

Function be like:

@api\_view(['POST'])

def TaskUpdate(request,pk):

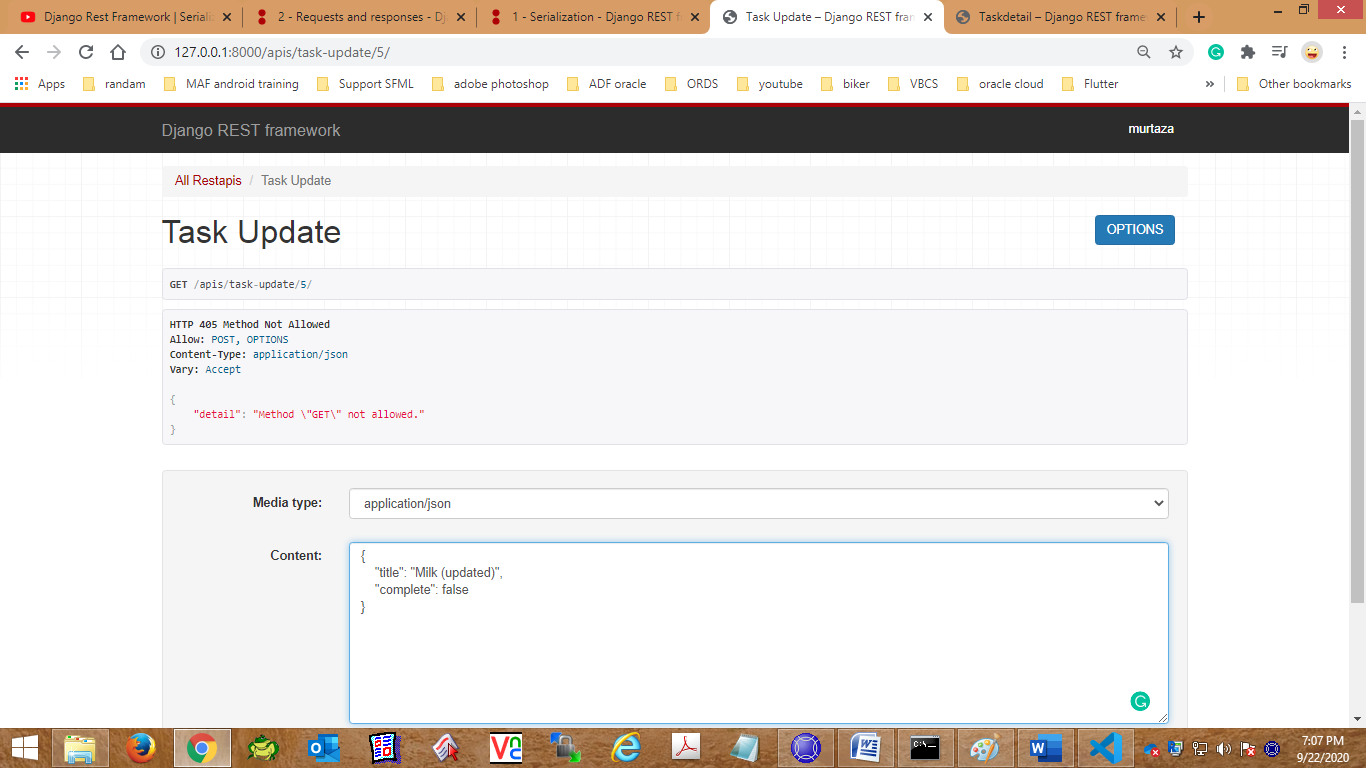
    task=Task.objects.get(id=pk)

    serialize=TaskSerializer(instance=task,data=request.data)

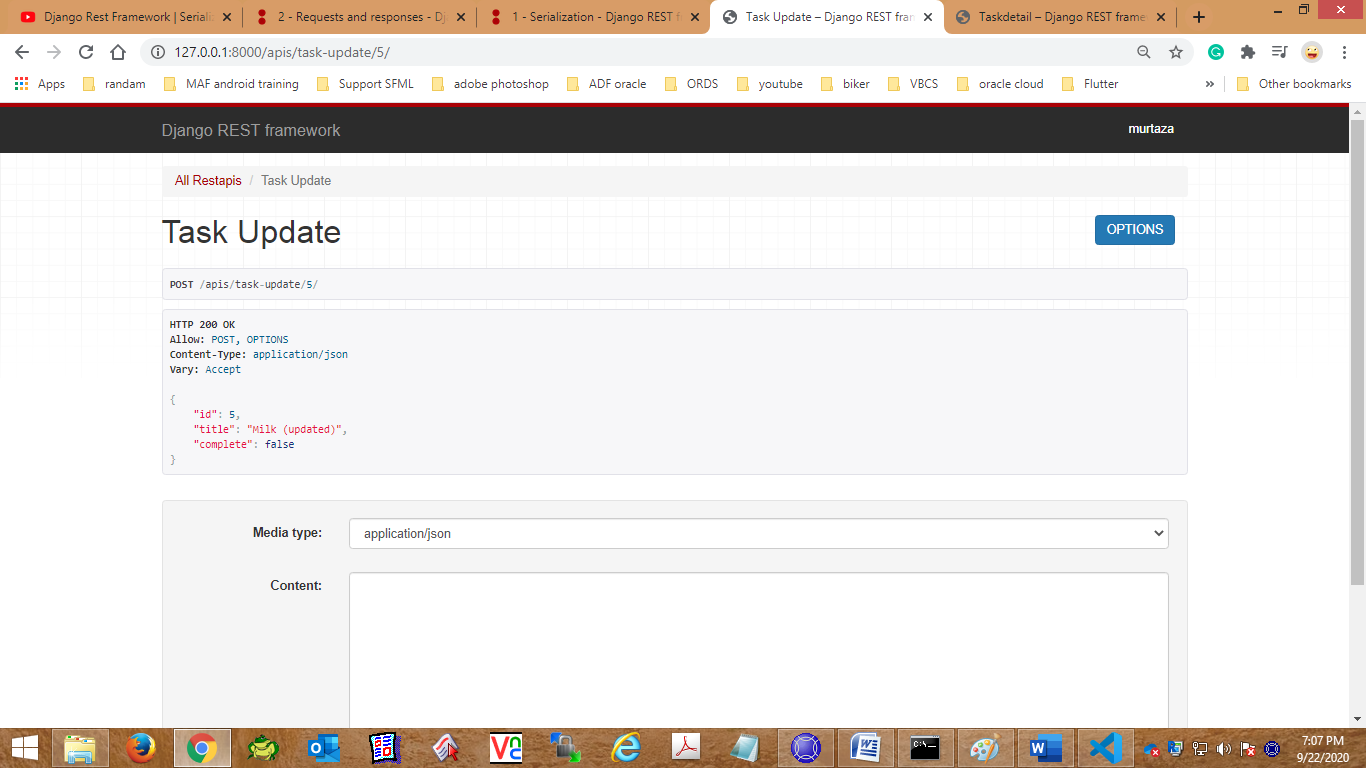
    if serialize.is\_valid():

        serialize.save()

    return Response(serialize.data)



Response:



# For task delete

url be like

path('task-delete/<str:pk>/', views.Taskdelete, name='Task-delete'),

Funtion be like.

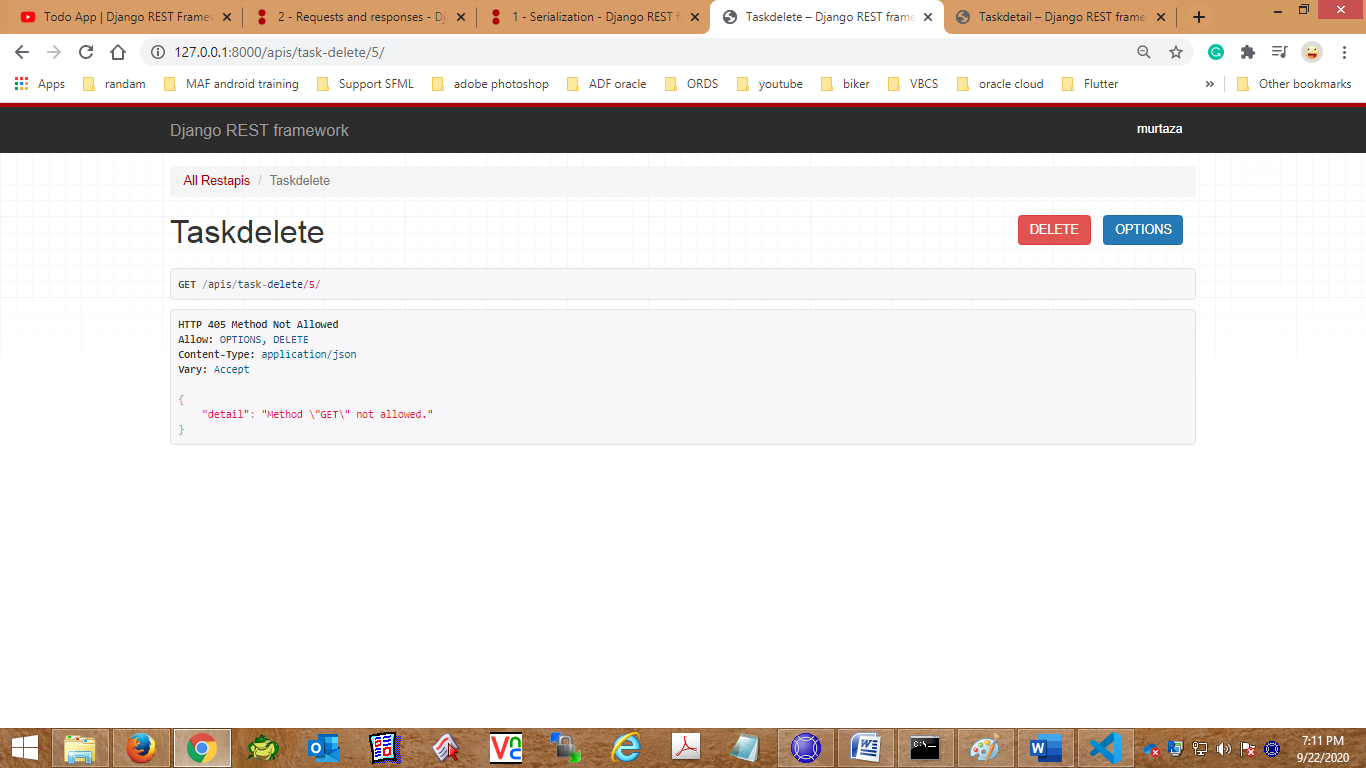
@api\_view(['DELETE'])

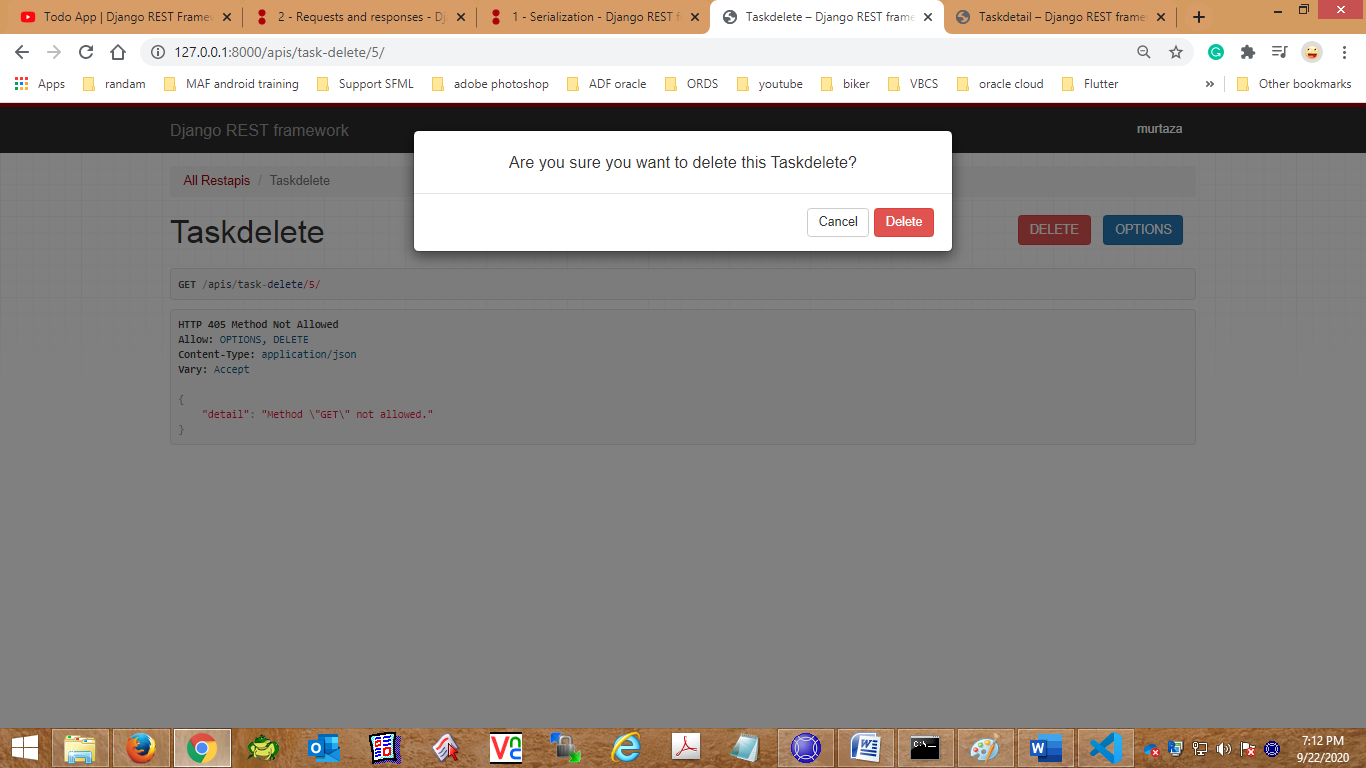
def Taskdelete(request,pk):

    task=Task.objects.get(id=pk)

    task.delete()

    return Response(serialize.data)





Response:

