Django Rest Framework:

WebServices:

→ A WebService is a collection of standards or protocals for exchanging information between two different Applications

→ A Webservice is a Software System for machine to machine communication

→ We can define mainly two types of WebServices they are:

① Soap WebServices

② Restful Web Services

① Soap WebService:

Soap → simple object access protocol

→ Soap is a XML based protocol for accessing Webservices

→ Soap is a platform independent and language independent

→ In soap Webservices, We should follow the Security Standards [WS Security] While building WebServices

→ the main limitation of soap based WebServices are:

a) It will give the data in the form of XML format and parsing that XML file slower and more network band width is required to transfer the XML file.

b) Implementation of soap based WebServices is difficult.

② Restful WebService :

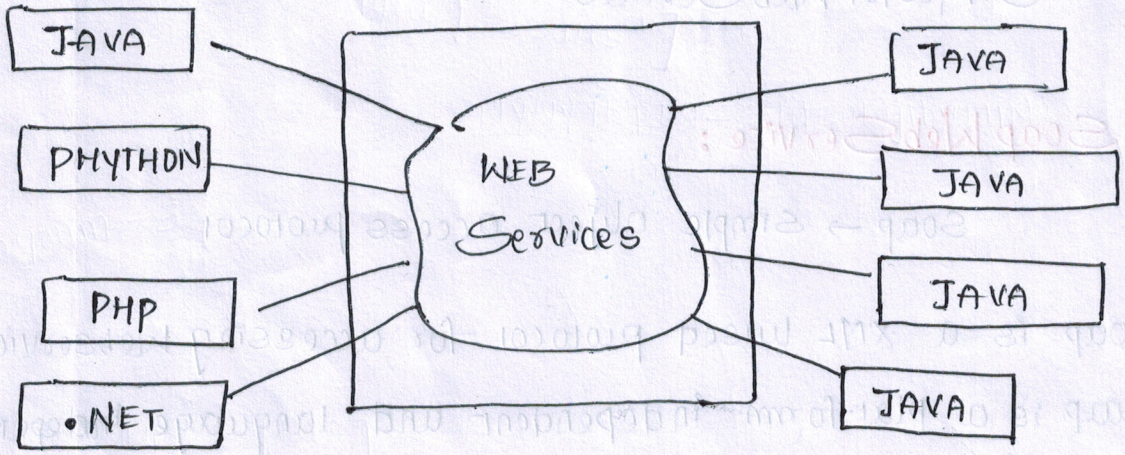
REST → Representional State Transfer

→ Rest is not a protocol but it is an Architectural Style

→ Restful Web services are faster because there is no strict Specification and also less network bandwidth is required

→ Restful WebServices Support Various Varities of data fomat like Text, XML and JSON etc

→ Restful WebServices can use soapbased Web Services but Soap based WebServices can't use Restful Web Services



→ the Application which is providing the WebServices is known as a "WebService Provider"

→ the Application which is accessing the data through the WebService is known as "WebService Consumer"

→ We can access the data from the Website into the python program by sending the request to the API's URI of the Website.

→ We can send the request to the Website through the python pgm by using functions of request module

→ request module is a external module and we have to install that module by using pip

```
c:\python36\Scripts > pip install requests
```

Program :1

```
import requests
url = 'https://api.speedcurve.com/v1/urls?days=1
key = 'cn685aw5tj0i4753d30sbmtpsxy8q'
password = 'Any_password'
response = requests.get(url, auth=(key, password))
d=(response.json())
Print(d)
```

Program-2:

```
import requests
import json
url = 'https://api.speedcurve.com/v1/urls?days=1'
key = 'cn685aw5tj0i4753d30sbmtpsx8q'
password = 'Any
response = requests.get(url, auth = (key, password))
d=(response.json())
fil = open("mydata.json", "a")
for K,V in d.items():
    for p in v:
        url = 'https://api.speedcurve.com/v1/urls?site_id=
            +str(p.get("site_id"))
```

```
        res = requests.get(url, auth=(key, password))
        fres = (res.json())
        json.dump(fres, fil, indent=True)
fil.close()
```

Program -3

```
import requests
sample_data = ['eid', 'campaign', 'type', 'prospects', 'advertiser',
               'kpi-goal']

data_dict = { }
adroll_auth = ('hkoppuravuri@nerdwallet.com', 'Happy-1234')
url1 = 'http://services.adroll.com/api/v1/report/campaign/
       get?apikey=2WYWY7WPTAZC9Q5ORGIHVBHNYOE2JUJZ'
r = requests.get(url1, auth=adroll_auth)
data = r.json()
#print(data)
mydata = data["results"]
outlist = [ ]
for p in mydata:
    # print(p)
    Ourdict = { }
    for q in sample_data:
        if q in p
            Ourdict[q] = P.get(q)
    outlist.append(Ourdict)
Print(outlist)
```

In order to build the RestApi in the Website of django framework We use the django Rest-framework

Installation of Django Rest FrameWork:

Step1: Download and Install python software

Step2: create a virtual environment for python software

```
c:\python36>Pip install Virtualenv
```

create a folder with name djangorestapi in the c-drive and execute the following command

```
c:\djangorestapi>c:\python36\Scripts\Virtualenv myenv
```

Step3: Install django framework in Virtual environment

```
c:\djangorestapi>myenv\Scripts\activate
```

```
(myenv) c:\djangorestapi>pip install django
```

Step4: `(myenv) c:\djangorestapi>pip install django rest-framework`

Any user defined class which is extending rest_framework.serializers.Serializer class is known as a Serializer class

Serializer class allows to convert the python objects [Querysets, model objects] in the form of native python datatypes

If the data is in the form of native python datatypes format then we can easily represent the data in the form of any MIME format [JSON, XML, --- etc]

Serializer class defining is similar to that of Form class implementation in django framework.

```
from datatime import datatime
from rest_framework import serializers
class CommentSerializer(serializers.Serializer):
    email = serializers.EmailField()
    content = serializers.CharField(max_length=200)
    created = serializers.DateTimeField()
```

→ We can define n number of Serializer classes in our Application.

→ All Serializer classes related to an Application should be defined with in serializers.py file of application.

→ Django rest-framework internally contains one predefined application called rest_framework.

→ rest_framework internally contains the serializers module and that module contains the Serializer class.

→ By extending Serializer class we can create the user defined Serializer classes.

→ rest_framework application contains render module and that render module contains different classes by using those classes we can represent the python native data in the form of different MIME format.

```
c:\django restapi > myenv \ Scripts \ activate
(myenv) c:\django rest api > django-admin startproject myrestproj1
(myenv) c:\django rest api > myrestproj1 > python manage.py startapp
                                                    myrestapp1
```

→ open the settings.py file of project and add following application in the INSTALLED_APPS :

```
['rest_framework',
'myrestapp1',]
```

→ open the views.py file of application and add the following code

```
from django.shortcuts import render
from datetime import datetime.

class Comment :

    def __init__(self, email, content, created = None) :
        self.email = email
        self.content = content
        self.created = created or datetime.now( )
```

→ Create Serializers.py file in application and add code :

```
from rest_framework import Serializers
class CommentSerializer(serializers.Serializer):
    email = Serializers.EmailField()
    content = Serializers.CharField(max_length=200)
    created = Serializers.DateTimeField()
```

```
C:\djangorestapi\myrestproj1> python manage.py shell
```

```
from myrestapp1.views import comment
from myrestapp1.Serializers import CommentSerializer
comment = Comment(email='abc@example.com', content='foo bar')
Serializer = CommentSerializer(comment)
Serializer.data
from rest_framework.renderers import JSONRenderer
json = JSONRenderer().render(Serializer.data)
json
```

Project - 1

Click on file → new project → Django → enter project name : "Restproj2" → Expand project interpreter → select existing interpreter → Expand more settings → Enter application name as : "restapp2" → Create → OK

→ Click on Settings.py and add in Installed_Apps

```
Installed_Apps = [ 'rest_framework', ]
```

→ open Views.py file of Application and add the code :

```
from datetime import datetime
from .serializers import CommentSerializer
from rest_framework.renderers import JSONRenderer
class Comment:
    def __init__(self, email, content, created=None):
        self.email = email
        self.content = content
        self.created = created or datetime.now()

def myapi(request):
    comment = Comment(email='abc@example.com', content=
                      'foo bar')
    Serializer = CommentSerializer(comment)
    return JSONRenderer().render(Serializer.data)
```

→ Right click on Application → new → python file → Enter the name as : "Serializers" → ok

```
from rest_framework import serializers
class CommentSerializer(serializers.Serializer):
    email = serializers.EmailField()
    content = serializers.CharField(max_length=200)
    created = serializers.DateTimeField()
```

→ Click on urls.py file of project and add the following import stmt and url pattern

```
from django.conf.urls import url
from restapp2 import views

url(r'^restapp2/display', views.myapi, name='myapi')
```

→ Start the Server and type the url

```
http://localhost:8000/display
```

We can send request to the webApplication by using ~~http~~ httpie

```
C:\Python36\Scripts > pip install httpie
```

open the command prompt and type following command

```
C:\Users\LOKESH > http http://127.0.0.1:8000/restapp2/display
```

→ We can send the request to api by using python program:

for restapp2

```
import requests
url = 'http://127.0.0.1:8000/restapp2/display'
response = requests.get(url)
d = (response.json())
print(d)
for k,v in d.items():
    print(k,v)
```

Project : 3

→ Serializers working with Modelclass:

Click on File → newproject → Django → Enter name as 'restproj3' → Expand project more settings → Enter Application name as : 'restapp3' → Create → OK

→ Open settings.py file of project and add the following application name. in Installed_Apps

```
['rest_framework',]
```

→ Open Serializers.py file of Application and add the following code:

```
from rest_framework import serializers
class ProductSerializer(serializers.Serializer):
    pid = serializers.IntegerField()
    pname = serializers.CharField(max_length=20)
    pcost = serializers.DecimalField(max_digits=10, decimal_places=4)
    pmfdt = serializers.DateField()
    pexpdt = serializers.DateField()
```

→ open views.py file of Application and add the following code:

```
from django.shortcuts import render
from .models import Product
from django.http import HttpResponse
from .serializers import ProductSerializer
from rest_framework.renderers import JSONRenderer
def input(request):
    return render(request, 'input.html')
```

```
def insert(request):
    pid1 = int(request.GET['t1'])
    pname1 = request.GET['t2']
    pcost1 = float(request.GET['t3'])
    pmfdt1 = request.GET['t4']
    pexpdt1 = request.GET['t5']
    f = product(pid=pid1, pname=pname1, pcost=pcost1,
                pmfdt=pmfdt1, pexpdt=pexpdt1)
    f.save()
    return render(request, 'links.html')

def display(request):
    recs = product.objects.all()
    return render(request, 'display.html', {'records': recs})

def productapi(request):
    data = product.objects.all()
    serializer = ProductSerializer(data, many=True)
    return HttpResponse(JSONRenderer().render(serializer.data))
```

→ Right click on Application → new → python file → enter name : urls. → OK and add the code :

```
from django.conf.urls import url
from . import views
app_name = 'restapp3'
urlpatterns = [
    url(r'^$', views.input, name='input'),
    url(r'^insert$', views.insert, name='insert')
    url(r'^display$', views.display, name='display')
    url(r'^productapi$', views.productapi, name='productapi'),
```

→ Right click on templates → new → HTML file → enter as "input" → ok → and add following code.

```
<form action = "./insert" method = get >
Enter pid : <input type = "text" name = "t1" ><br>
Enter pname : <input type = "text" name = "t2" ><br>
Enter pcost : <input type = "text" name = "t3" ><br>
Enter pmfdt : <input type = "text" name = "t4" ><br>
Enter expdt : <input type = "text" name = "t5" ><br>
<input type = "submit" value = "submit" >
</form>
</body>
```

→ Right click on templates → new → HTML file → Enter name as: links → ok and add the following code:

```
<h1> product inserted successfully </h1><br>
<a href = "/restapp3"> click here to insert another product </a><br>
<a href = "/restapp3/display"> click here to display the products </a>
```

→ Right click templates → new → HTML file → Enter name as: display → ok and add the following code:

```
<table border = "1" >
<tr>
    <th> pid </th>
    <th> pname </th>
    <th> pcost </th>
    <th> pmfdt </th>
    <th> perpdt </th>
</tr>

{% for rec in records %}
<tr>
    <td> {{ rec.pid }} </td>
    <td> {{ rec.pname }} </td>
    <td> {{ rec.pcost }} </td>
    <td> {{ rec.pmfdt }} </td>
    <td> {{ rec.perpdt }} </td>
</tr>
```

```
{ % endfor % }
</table>
<a href="/restapp3" > click here to insert another product </a>
</body>
```

→ open project urls.py file and add following import stmts and urlpattern.

```
import from django.conf.urls import url
       from django.conf.urls import include
[url pattern s = [
    url (r'^ restapp3 /', include ('restapp3.urls')),
                ]
```

→ Click on tools → Run manage.py task and execute following commands

→ makemigrations

→ migrate

→ we can send the request to the api by using python program :

```
import requests
import json
url = 'http://localhost:8000/restapp3/productapi'
response = requests.get(url)
q = (response.json())
# print (d)
x = [ ]
for p in q:
        for a,b in p.items():
                print (a,b)
                if a == "pcost":
                        x.append(b)
tot = 0
for p in x:
    tot = tot + float(p)
print(tot)
```

→ open models.py file of application of testapp3 and add the following code:

```
from django.db import models
class Product(models.Model):
    pid = models.IntegerField(primary_key=True)
    pname = models.CharField(max_length=20)
    pcost = models.DecimalField(max_digits=10,
                                decimal_places=4)
    pmfd = models.DateField()
    pexpdt = models.DateField()
```

Model Serializer:

→ If we want to define Serializer class, with model class attributes we can use ModelSerializers, Instead of defining the Normal Serializer class.

→ Any user define class which is extending rest_framework.Serializers.ModelSerializer class is known as ModelSerializer.

→ We can store the data into the db table directly by calling save() method on the ModelSerializer class object

→ The data which is given by the render() of JSONRenderer class object is taken by the HttpResponse class and gives the data in form of text-format.

→ In order to gives the data in the form of json-format we can specify the content_type parameter value as a 'application/json'.

Project -4

click on File → new project → Django → Enter name : 'restproj4' →
Expand project more settings → Enter Application name : restapp4
→ Create → OK

→ open settings.py file of project and add the following Application name in Installed_Apps :

```
['rest_framework']
```

→ open models.py file of application and add the following code :

```
from django.db import models
class Product(models.Model):
    pid = models.IntegerField(primary_key = True)
    pname = models.CharField(max_length = 20)
    pcost = models.DecimalField(max_digits = 10,
                                decimal_places = 4)
    pmfd = models.DateField()
    pexpdt = models.DateField()
```

→ Open the Serializers.py of Application, and add the following code :

```
from rest_framework import Serializers
from .models import Product
class ProductSerializer(Serializers.ModelSerializer):
    class Meta:
        model = Product
        fields = ('pid', 'pname', 'pcost', 'pmfd', 'pexpdt')
```

→ open views.py file of Application and add the follow

```
from django.shortcuts import render
from .models import Product
from django.http import HttpResponse
from .serializers import ProductSerializer
from rest_framework.renderers import JSONRenderer
class JSONResponse(HttpResponse):
    def __init__(self, data, **kwargs):
        content = JSONRenderer().render(data)
        kwargs['content_type'] = 'application/json'
        super(JSONResponse, self).__init__(content,
                                           **kwargs)

def input(request):
    return render(request, 'input.html')

def insert(request):
    pid1 = int(request.GET['t1'])
    pname1 = request.GET['t2']
    pcost1 = float(request.GET['t3'])
    pmfd1 = request.GET['t4']
    pexpdt1 = request.GET['t5']
    f = Product(pid=pid1, pname=pname1,
                pcost=pcost1, pmfd=pmfd1, pexpdt=pexpdt1)
    f.save()
    return render(request, 'links.html')
```

```
def display(request):
    recs = Product.objects.all()
    return render(request,'display.html',{'records':recs})

def productapi(request):
    data = product.objects.all()
    Serializer = ProductSerializer(data,many=True)
    return JsonResponse(Serializer.data)
```

→ Right click on Application → new → python file → entername: url
→ ok and add the code:

```
from django.conf.urls import url
from .import views
app_name ='restapp4'
urlpatterns = [
url(r'^$', views.input, name ='input'),
url(r'^insert$', views.insert, name ='insert')
url(r'^display$', views.display, name ='display')
url(r'^productapi$', views.productapi, name='productapi')
```

→ Right click on templates → new → HTML file → enter as "input"
→ ok → and add following code:

```
<form action = "./insert" method =get>
Enter pid: <input type = "text" name = "t1" ><br>
Enter pname : <input type = "text" name = "t2" ><br>
Enter pcost : <input type = "text" name ="t3"><br>
Enter pmfd : <input type ="text" name ="t4"><br>
Enter expdt : <input type = "text" name ="t5"><br>
    <input type = "submit" value ="Submit">
</form>
```

→ Right click on templates → new → HTML-file → Enter name : links
→ ok and add the following code :

```
<h1> Product inserted success-fully </h1><br>
<a href = "/restapp4" > click here to another product </a><br>
<a href = "/restapp4/display" > click here to display products </a>
```

→ Right click templates → new → HTML-file → Enter name : display
→ ok and add the following code :

```
<table border = "1">
<tr>
    <th> pid </th>
    <th> pname </th>
    <th> pcost </th>
    <th> pmfd </th>
    <th> pexpdt </th>
</tr>
{ % for rec in records % }
<tr>
    <td> {{ rec.pid }} </td>
    <td> {{ rec.pname }} </td>
    <td> {{ rec.pcost }} </td>
    <td> {{ rec.pmfd }} </td>
    <td> {{ rec.pexpdt }} </td>
{ % end-for % }
</table>
<a href = "/restapp4"> click here to insert another product </a>
</body>
```

→ open project urls.py file and add following import stmts and urlpattern

```
from django.conf.urls import url
from django.conf.urls import include

urlpatterns = [

    url(r'^restapp4/', include('restapp4.urls')),

]
```

→ click on tools → Run manage.py task and execute following commands :

```
>>> makemigrations
>>> migrate.
```

APIVIEW:

→ APIView is a predefined class which is defined in rest_framework.views module

→ whenever our class is extending APIView class that class methods can return the Response object

→ Response class is a predefined class which is defined in rest_framework.

→ Response class object take the Serializable class object data and it will return the data to the enduser in the form of JSON format.

Project-5

Click on File → newproject → Django → Enter name : 'restproj5' → Expand project more settings → Enter Application name : 'restapp4' → create → OK

→ Open settings.py file of the project and add the following code:

Application name in Installed_Apps:

```
['rest_framework']
```

→ Open models.py file of Application and add the following code:

```
from django.db import models
class Product(models.Model):
    pid = models.IntegerField(primary_key = True)
    pname = models.charfield(max_length =20)
    pcost = models.DecimalField(max_digits =10,
                                decimal_places =4)
    pmfd = models.DateField( )
    pexpdt=models.DateField( )
```

→ Open the Serializers.py of Application and the following code:

```
from rest_framework import serializers
from .models import Product
class ProductSerializer(serializers.ModelSerializer):
    class Meta:
        model = Product
        fields = ('pid','pname','pcost','pmfd','perpdt')
```

→ Open Views.py file code Application and add the following code:

```
from django.shortcuts import render
from .models import Product
from .serializers import ProductSerializer
from rest_framework.views import APIView
from rest_framework.response import Response
def input(request):
    return render(request,'input.html')
def insert(request):
    pid1 = int(request.GET['t1'])
    pname1 = request.GET['t2']
    pcost1 = float(request.GET['t3'])
    pmfd1 = request.GET['t4']
    perpdt1 = request.GET['t5']
    f = Product(pid=pid1,pname=pname1,pcost=pcost1,
                pmfd=pmfd1,perpdt=perpdt1)
    f.save()
    return render(request,'links.html')
```

```
def display(request):
    recs = Product.objects.all()
    return render(request, 'display.html', {'records': recs})

class ProductList(APIView):
    def get(self, request, format=None):
        products = Product.objects.all()
        Serializer = ProductSerializer(products, many=True)
        return Response(Serializer.data)
```

→ Right click on Application → new → python file → enter name: urls
→ ok and add the code:

```
from django.conf.urls import url
from . import views
app_name = 'restapps'
urlpatterns = [
url(r'^$', views.input, name='input'),
url(r'^insert$', views.insert, name='insert')
url(r'^display$', views.display, name='display')
url(r'^productapi/$', views_productList.as_view()),
```

→ Right click on templates → new → HTML file → enter as: "input"
→ ok → and add following code:

```
<form action="./insert" method=get>
Enter pid : <input type="text" name="t1"><br>
Enter pname : <input type="text" name="t2"><br>
Enter pcost : <input type="text" name="t3"><br>
Enter pmfd : <input type="text" name="t4"><br>
Enter pexpdt : <input type="text" name="t5"><br>
<input type="submit" value="submit">
</form>
```

→ Right click on templates → new → HTML code → Enter name: links → ok and add the following code:

```
<h1> Product inserted Successfully </h1><br>
<a href = "/restapp5">Click here to another product </a><br>
<a href = "/restapp5/display">click here to display products</a>
```

→ Right click on templates → new → HTML file: display → ok and add the following code:

```
<table border="1">
<tr>
    <th> pid </th>
    <th> pname </th>
    <th> pcost </th>
    <th> pmfd </th>
    <th> pexpdt </th>
</tr>
{% for rec in records %}
<tr>
    <td> {{rec.pid}} </td>
    <td> {{rec.pcost}} </td>
    <td> {{rec.pname}} </td>
    <td> {{rec.pmfd}} </td>
{% endfor %}
</table>
<a href="/restapp5"> click here to insert another product</a>
</body>
```

→ open project urls.py file and add the following import statements and url pattern :

```
from django.conf.urls import url
from django.conf.urls import include

urlpatterns = [
    url(r'^restapps/', include('restapps.urls')),
]
```

→ Click on tools → Runmanage.py task and execute the following commands :

```
>>> makemigrations
>>> migrate
```

LISTAPIView class :

LISTAPIView class is a predefined class which is defining generics module of rest_framework.

→ Whenever our class is extending LISTAPIView class we need not to define the 'get' method logic

→ Within the subclass of LISTAPIView class if we provide the Queryset object and Serializer class name then internally implement the serialization on Queryset objects and gives the response in the form of json format.

Project : 6

Click on File → new project → Django → Enter name : 'restproj6'
→ Expand project more settings → Enter Application name : 'restapp6'
→ Create → OK

→ Is same as project : 5 → Except : views.py file

settings.py → same as project 5
models.py → same as project 5
Serializers.py → same as project 5
templates(~~urls.py~~, input.html, link.html, display.html) → same as proj-5
urls.py of Application and project → same as project 5

→ Open views.py of Application : (modifications)

```
from rest_framework import generics  ⟶ extra

class ProductList (generics.LISTAPIView):
    queryset = product.objects.all()
    serializer_class = producr Serializer
```

the disadvantage of ListAPIView class is we can use only "get" request and cannot access other requests. Hence we go for API View only.

Next

Handling 'GET' and 'POST' Requests: Project: 7

Click on File → newproject → Django → Enter name: 'restproj7' → Expand project more settings → Enter Application name: 'restapp7' → Create → OK

→ open the settings.py file of the project and add the following Application name in Installed_Apps: code:

```
['rest_framework']
```

→ open models.py file of Application and add the following code:

```
from django.db import models
class Product(models.Model):
    pid = models.IntegerField(primary_key=True)
    pname = models.CharField(max_length=20)
    pcost = models.DecimalField(max_digits=10,
                                decimal_places=4)
    pmfd = models.DateField()
    perpdr = models.DateField()
```

→ open Serializers.py of Application and add the following code:

```
from rest_framework import Serializers
from .models import Product
class ProductSerializer(serializers.ModelSerializer):
    class Meta:
        model = product
        fields = ('pid', 'pname', 'pcost', 'pmfd', 'perpdr')
```

→ open views.py file of Application and add the following code:

```
from django.shortcuts import render
from .models import product
from rest_framework import generics
from .serializers import productSerializer
from rest_framework.renderers import JSONRenderer
from rest_framework.views import APIView
from rest_framework.response import Response.

def inpur(request):
    return render(request, 'inpur.html')

def insert(request):
    pid1 = int(request.GET['t1'])
    pname1 = request.GET['t2']
    pcost1 = float(request.GET['t3'])
    perpdt1 = request.GET['t5']
    pmfd = request.GET['t4']
    f = Product(pid=pid1, pname=pname1, pcost=pcost1, pmfd=pmfd1,
                                                    perpdt=perpdt)
    f.save()
    return render(request, 'links.html')

def display(request):
    recs = product.objects.all()
    return render(request, 'display.html', {'records': recs})

class ProductList(APIVIEW):
    def get(self, request, format=None):
        products = Product.objects.all()
        serializer = productSerializer(products, many=True)
        return Response(serializer.data)
    def post(self, request, format=None):
        serializer = ProductSerializer(data=request.POST)
```

```
if serializer.is_valid():
    Serializer.save()
    return Response(Serializer.data)
return Response(Serializer.errors)
```

→ Right click on Application → new → python file → Enter name : urls
→ ok → and add the following code :

```
from django.conf.urls import url
from . import views
app_name = 'restapp1'
urlpatterns = [
    url(r'^$', views.input, name='input')
    url(r'^display$', views.display, name='display')
    url(r'^productapi$', views.ProductList.as_view()),
]
```

→ Right click on templates → new → HTML file → Enter name : "input"
→ ok and add following code :

```
<form action="/productapi" method="post">
    Enter pid : <input type="text" name="pid"><br>
    Enter pname : <input type="text" name="pname"><br>
    Enter pcost : <input type="text" name="pcost"><br>
    Enter pmfd : <input type="text" name="pmfd"><br>
        <input type="submit" value="submit">
</form>
```

```
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors)
```

→ Right click on Application → new → python file → Enter name : urls
→ ok → and add the following code :

```
from django.conf.urls import url
from . import views
app_name = 'restapp1'
urlpatterns = [
    url(r'^$', views.input, name='input')
    url(r'^display$', views.display, name='display')
    url(r'^productapi$', views.ProductList.as_view()),
]
```

→ Right click on Templates → new → HTML file → Enter name : "input"
→ ok and add following code :

```
<form action="./productapi" method="post">
    Enter pid : <input type="text" name="pid"><br>
    Enter pname : <input type="text" name="pname"><br>
    Enter pcost : <input type="text" name="pcost"><br>
    Enter pmfd : <input type="text" name="pmfd"><br>
        <input type="submit" value="Submit">
</form>
```

→ Right click on templates → new → HTML file → Enter name: links → ok and add the following code:

```
<h1> Product Inserted Succesfully </h1><br>
<a href = "/restapp7" > click here to insert another product </a><br>
<a href = "/restapp7/display > click here to display the product </a>
<
```

→ Right click on templates → new → HTML file : display → ok and add the following code:

```
<table border = "1">
<tr>
    <th> pid </th>
    <th> pname </th>
    <th> pcost </th>
    <th> pmfd </th>
    <th> pexpdt </th>
</tr>

{% for rec in records %}

<tr>
    <td> {{ rec.pid }} </td>
    <td> {{ rec.pcost }} </td>
    <td> {{ rec.pname }} </td>
    <td> {{ rec.pmfd }} </td>
    <td> {{ rec.pexpdt }} </td>

{% endfor %}

</table>
<a href = "/restapp7" > click here to insert Another product </a>
```

→ open project urls.py file and add the following import stmts and url patterns:

```
from django.conf.urls import url
from django.conf.urls import include.

url patterns = [
    url (r'^ restapp1/', include ('restapp1.urls')),
]
```

→ click on tools → Run manage.py task and execute the following commands:

```
>>> make migrations
>>> migrate.
```

→ Instead of getting all the products information whenever we send the request to API we have to get particular product Information which product id is submitted by the Enduser.

→ We develop the Seperate Viewclass to get particular product id information and define the API url for that product class.

Project 8:

Click on File → newproject → Django → Entername : 'restproj8'
Expand project more settings → Enter Application name : 'restapp8' → click on create → ok.

→ Open the settings.py file of the project and add the Application name in Installed_Apps :

```
['rest_framework']
```

→ Open models.py file of Application and add the following code:

```
from django.db import models
class Product(models.Model):
    Pid = models.IntegerField(primary_key=True)
    Pname = models.CharField(max_length=20)
    Pcost = models.DecimalField(max_digits=10, decimal_places=4)
    Pmfd = models.DateField()
    Pexpdt = models.DateField()
```

→ open Serializers.py of Application and Add the following code:

```
from rest_framework import Serializers
from .models import product
class ProductSerializer(serializers.ModelSerializer):
    class Meta:
        model = product
        fields = ('pid', 'pname', 'pcost', 'pmfd', 'pexpdt')
```

→ open views.py file of Application and add the following code:

```
from django.shortcuts import render
from .models import product
from rest_framework import generics
from .serializers import ProductSerializer
from rest_framework.renderers import JSONRenderer
from rest_framework.response import Response
from rest_framework.views import APIView.

def input(request):
    return render(request, 'input.html')
def insert(request):
    pid1 = int(request.GET['t1'])
    pname1 = request.GET['t2']
    pcost1 = float(request.GET['t3'])
    pmfd1 = request.GET['t4']
    pexpd1 = request.GET['t5']
    f = Product(pid=pid1, pname=pname1, pcost=pcost1, pmfd=pmfd1, pexpdt=
                                                        pexpd1)
    f.save()
    return render(request, 'links.html')
def display(request):
    recs = Product.objects.all()
    return render(request, 'display.html', {'records': recs})
```

```
Class ProductList(APIView):
    def get(self, request, format=None):
        products = product.objects.all()
        Serializer = ProductSerializer(products, many=True)
        return Response(Serializer.data)
    def post(self, request, format=None):
        Serializer = ProductSerializer(data=request.POST)
        if Serializer.is_valid():
            Serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors)
def inpurpid(request):
    return render(request, 'inpurpid.html')
Class ProductDetail(APIView):
    def get(self, request, format=None):
        pid1 = int(request.GET['pid'])
        prod = product.objects.filter(pid=pid1)
        prod.delete()
        Serializer = ProductSerializer(prod, many=True)
        return Response(serializer.data)
```

→ Right click on Application → new → python-file → Enter name : urls
→ OK and add the following code :

```
from django.conf.urls import url
from . import views
app_name = 'restapp'
urlpatterns = [
    url(r'^$', views.inpur, name='inpur'),
    url(r'^display$', views.display, name='display'),
    url(r'^productapi$', views.ProductList.as_view()),
    url(r'^inpurpid$', views.inpurpid, name='inpurpid'),
    url(r'^product$/', views.ProductDetail.as_view()),
]
```

→ Open the input.html and add the following code:

```
<body bgcolor = "#00ffff">
<form action = "./productapi" method = "post">
Enter pid :<input type = "text" name = "pid"><br>
Enter pname:<input type = "text" name = "pname"><br>
Enter pcost :<input type = "text" name = "pcost"><br>
Enter pmfd : <input type = "text" name = "pmfd"><br>
Enter pexpdt : <input type = "text" name = "pexpdt"><br>
    <input type = "submit" value = "submit">
</form>
</body>
```

→ Open the inputid.html and add the following code:

```
<!DOCTYPE html>
<html lang = "en">
<head>          </head>
<body bgcolor = "aqua">
<form action = "./products" method = "get">
  Enter pid : <input type = "text" name = "pid"><br>
  <input type = "submit" value = "get">
</form>
</body>
```

→ Open the links.html and add the following code:

```
<body bgcolor = "gray">
<h1> product inserted successfully </h1><br>
<a href = "/restapps"> click here to insert another product </a><br>
<a href = "/restapps/display> click here to display the products </a>
</body>
```

→ open the display.html and add the following code :

```
<body bgcolor = "gray">
<table border = "1">
<tr>
    <th> pid </th>
    <th> pname </th>
    <th> pcost </th>
    <th> pmfd </th>
    <th> perpdt </th>
</tr>

{% for rec in records %}

<tr>
    <td> {{ rec.pid }} </td>
    <td> {{ rec.pname }} </td>
    <td> {{ rec.pcost }} </td>
    <td> {{ rec.pmfd }} </td>
    <td> {{ rec.perpdt }} </td>
</tr>

{% endfor %}

</table>
<a href = "/restapps"> click here to insert another product </a>
</body>
```

→ open project urls.py file and add the following code.
import stmts and url patterns :

```
from django.conf.urls import url, include
urlpatterns = [
    url (r'^ restapps/', include ('restapps.urls')),
]
```

→ click on tools → Runmanage.py task and execute the following commands :

```
makemigrations
migrate
```

API Authentication Token

→ In order to give the accessibility of the API for the authenticated users we use API Authentication token Mechanism.

In order to define the API authentication Service we use 'rest_framework.authtoken' application

→ We define the permissions to the API's by overriding the methods of 'rest_framework.permissions.Basepermission' class.

→ Within the views.py file we have to provide the permission class information with in view class

→ permission classes we should define with in permissions.py file of the Application.

→ Whenever we Send the request to the 'rest_framework.authtoken.views.obtain_auth_token View' generates the a admin user

→ By using the authentication token authorised people can access the API

Click on Create File → new project → Django → enter name as 'restproj9' → Expand project Interpreter → Select Existing interpreter → Expand more settings → Enter Application name as: 'authapp' → Create → OK.

→ open the settings.py file of project and add following apps to the Installed_Apps = [

```
'rest_framework',
'rest_framework.authtoken',  ]
```

and the authentication token class at the end of settings.py file.

```
REST_FRAMEWORK = {
        'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.TokenAuthenti
        cation' )

        }
```

→ open the views.py of Application and add the following code:

```
from django.shortcuts import render
from django.shortcuts import render
from .models import product
from django.http import HttpResponse
from .serializers import ProductSerializer
from rest_framework.renderers import JSONRenderer
from rest_framework.views import APIView
from rest_framework import generics
from rest_framework.response import Response
from .permissions import IsAdminOrReadOnly

def input(request):
    return render(request, 'input.html')

def insert(request):
    pid1 = int(request.GET['t1'])
    pname1 = request.GET['t2']
```

API Authentication Token

→ In order to give the accessibility of the API for the authenticated users we use API Authentication Token Mechanism.

In order to define the API authentication Service we use 'rest_framework.authtoken' application

→ We define the permissions to the API's by overriding the methods of 'rest_framework.permissions.Basepermission' class.

→ Within the views.py file we have to provide the permission class information with in view class

→ permission classes we should define with in permissions.py file of the Application.

→ Whenever we send the request to the 'rest_framework.authtoken.views.obtain_auth_token View' generates the a admin user

→ By using the authentication token authorised people can access the API

Click on Create File → new project → Django → enter name as 'restproj9' → Expand project Interpreter → Select Existing interpreter → Expand more settings → Enter Application name as: 'authapp' → Create → OK.

```
    pcost1 = float(request.GET['t3'])
    pmfd1 = request.GET['t4']
    perpd1 = request.GET['t5']
    f = Product(pid=pid1, pname=pname1, pcost=pcost1,
                pmfd=pmfd1, perpdt=perpd1)
    f.save()
    return render(request, 'links.html')

def display(request):
    recs = Product.objects.all()
    return render(request, 'display.html', {'records': recs})

class ProductList(generics.ListCreateAPIView):
    queryset = Product.objects.all()
    serializer_class = ProductSerializer
    permission_classes = (IsAdminOrReadOnly,)
```

```
class ProductList1(generics.RetriveUpdateDestroyAPIView):
    queryset = product.objects.all()
    serializer_class = ProductSerializer
    permission_classes = (IsAdminOrReadOnly,)
```

→ Open the permissions.py file of Application and add the following code:

```
from rest_framework.permissions import BasePermission,
                                       SAFE_METHODS
class IsAdminOrReadOnly(BasePermission):
    def has_permission(self, request, view):
```

```
if request.method in SAFE_METHODS:
    return True
else:
    return request.user.is_staff
```

→ open the models.py of Application and add the code:

```
from django.db import models
class Product(models.Model):
    pid = models.IntegerField(primary_key=True)
    pname = models.CharField(max_length=20)
    pcost = models.DecimalField(max_digits=10,
                                decimal_places=4)
    pmfd = models.DateField()
    perpdt = models.DateField()
```

→ open the Serializers.py of Application & add the code:

```
from rest_framework import serializers
from .models import Product
class ProductSerializer(serializers.ModelSerializer):
    class Meta:
        model = Product
        fields = ('pid','pname','pcost','pmfd','perpdt')
```

→ Input.html :

```
<body bgcolor="red">
<form action="/insert" method="post">
Enter pid :<input type="text" name="t1"><br>
Enter pname :<input type="text" name="t2"><br>
Enter pcost :<input type="text" name="t3"><br>
Enter pmfd :<input type="text" name="t4"><br>
Enter perpdt :<input type="text" name="t5"><br>
```

```
<input type="submit" value="Submit">
</form>
</body>
```

→ display.html :

```
<body bgcolor="gray">
<table border="1">
<tr>
    <th> pid </th>
    <th> pname </th>
    <th> pcost </th>
    <th> pmfd </th>
    <th> pexpdt </th>
</tr>
{% for rec in records %}
<tr>
    <td> {{ rec.pid }} </td>
    <td> {{ rec.pname }} </td>
    <td> {{ rec.pcost }} </td>
    <td> {{ rec.pmfd }} </td>
    <td> {{ rec.pexpdt }} </td>
</tr>
{% end-for %}
</table>
<a href="/authapp"> click here to insert another product</a>
</body>
```

→ links.html :

```
<body bgcolor="#00ffff">
<h1> product inserted Successfully </h1><br>
<a href="/authapp"> click here to insert another product </a><br>
<a href="/authapp/display"> click here to display the products </a>
</body>
```

urls.py of Application :

```
from django.conf.urls import url
from . import views
app_name = 'authapp'
urlpatterns = [
url(r'^$', views.input, name='input'),
url(r'^insert$', views.insert, name="insert"),
url(r'^display$', views.display, name='display'),
url(r'^products/$', views.productList.as_view()),
url(r'^products/(?P<pk>[0-9]+)$', views.productList1.as_view()),
]
```

urls.py of Project :

```
from django.conf.urls import url, include
from rest_framework.authtoken.views import obtain_auth_token
urlpatterns = [
    url(r'^api-token-auth/', obtain_auth_token),
    url(r'^authapp/', include('authapp.urls')),
]
```

Nested Serializers:

→ Using one serializer into other serializer is known as a Nested Serializer

→ In order to define the Nested Model Serializers we should have to define the relationships between Model classes.

→ While defining the Serializer class through model serializer class we have to define another Attribute which represents another model Serializer.

→ With in the model serializer we have to specify the attribute name in the fields which represents the another Serializer

→ Whenever we send the request to the Serializer API then automatically another serializers data will be displayed with in that attribute.

→ In the below Example, whenever we send the request to the Musician Serializer then it will display the all Musicians along with Musician Albums.

→ We can send the any type of request to the Nested Serializers and also we can implement Authentication Tokens if it is required.

Project: 10

Click on create File → new project → Django → Enter name: as 'restproj10' → Expand project interpreter → Select existing interpreter → Expand more settings → Enter Application name as: 'restapp10' → Create → OK

→ open the settings.py file of project and add the following to the Installed-Apps = [

```
'rest_framework',
'rest_framework.authtoken', ]
```

→ open the views.py of Application and the following code:

```
from
from django.shortcuts import render
from .models import *
from .serializers import *
from rest_framework import generics

class MusicianListView(generics.ListCreateAPIView):
    Queryset = Musician.objects.all()
    Serializer_class = MusicianSerializer

class MusicianView(generics.RetrieveUpdateDestroyAPIView):
    Serializer_class = MusicianSerializer
    Queryset = Musician.objects.all()

class AlbumListView(generics.ListCreateAPIView):
    Queryset = Album.objects.all()
    Serializer_class = AlbumSerializer

class AlbumView(generics.RetrieveUpdateDestroyAPIView):
    Serializer_class = AlbumSerializer
    Queryset = Album.objects.all()
```

→ open the models.py file and add the following code:

```
from django.db import models
from __future__ import unicode_literals
Class Musician (models.Model):
    first_name = models.CharField (max_length =50)
    last_name = models.CharField (max_length =50)
    Instrument = models.CharField (max_length= 100)

    def __unicode__ (self):
        return self.first_name
class Album (models.Model):
    artist = models.ForeignKey (Musician, on_delete = models.CASCADE,
                related_name = 'album_musician', null = True, blank=True)
    name = models.CharField (max_length =100)
    release_date = models.DateField ( )
    num_stars = models.IntegerField ( )
```

→ open the Serializers.py file and add the following code:

```
from .models import *
from rest_framework import serializers, fields

class AlbumSerializer (serializers.ModelSerializer):

    class Meta :
        model = Album
        fields = ('id', 'artist', 'name', 'release_date', 'num_stars')

class MusicianSerializer (serializers.ModelSerializer):
    album_musician = AlbumSerializer (read_only = True, many = True)

    class Meta :
        model = Musician
        fields = ('id', 'first_name', 'last_name', 'instrument', 'album_m-
                                                                  usician')
```

→ Open urls.py of Project and add the following code:

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import url
from restapp10 import views

urlpatterns = [

    url(r'^api/musicians/$', views.MusicianListView.as_view()),
url(r'^api/musicians/(?P<pk>\d+)/$', views.MusicianView.as_view()),

url(r'^api/albums/$', views.AlbumListView.as_view()),
url(r'^api/albums/(?P<pk>\d+)/$', views.AlbumView.as_v
                                                    iew()),

]
```

NEW Project

Click on create File → new project → Django → Enter name as 'restproj10' → Expand project interpreter → select existing interpreter → Expand more settings → Enter Application name as: 'restappReview' → create → OK.

→ open the settings.py file of project and add the following Apps in to the Installed_Apps = [

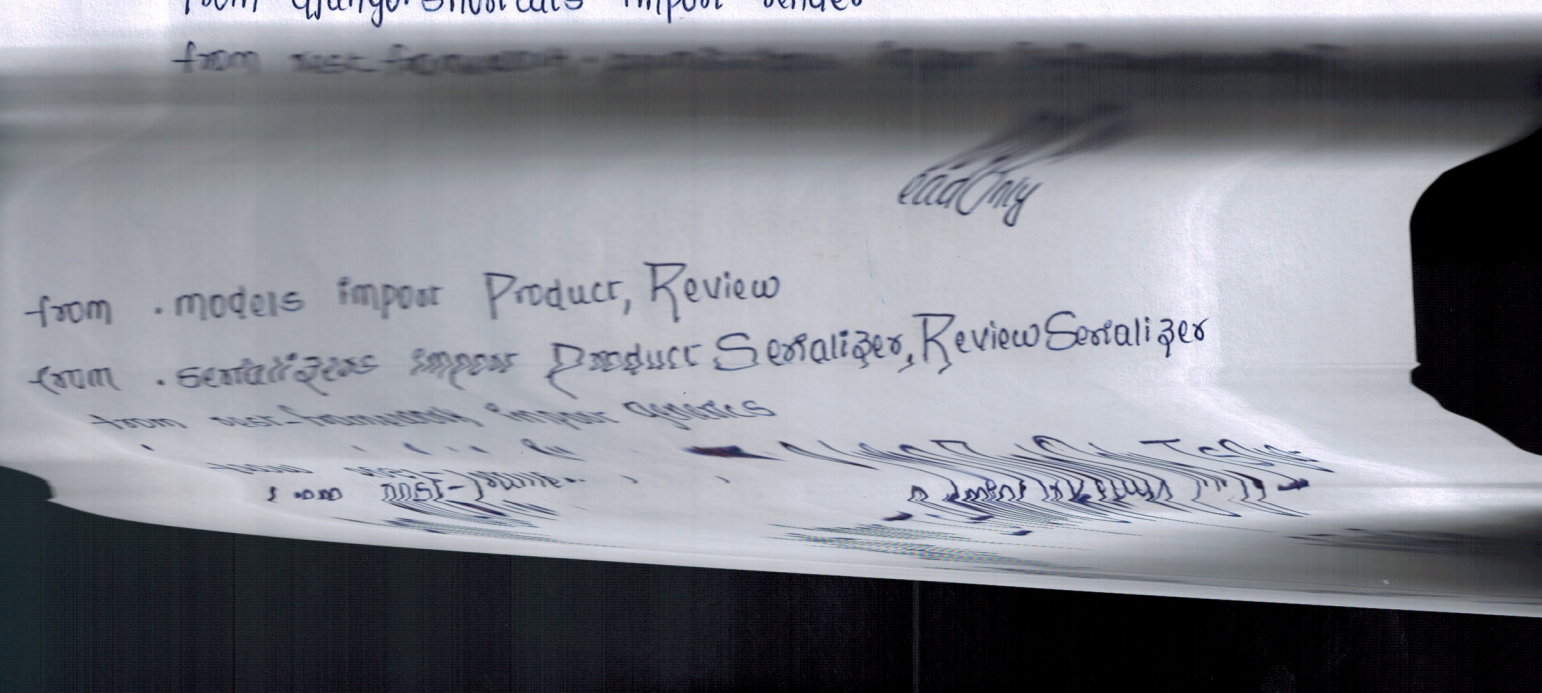
```
'rest_framework',
'rest_framework.authtoken', ]
```

and the authentication token class at the end of the settings.py file:

```
REST_FRAMEWORK = {
        'DEFAULT_AUTHENTICATION_CLASSES': (
'rest_framework.authentication.TokenAuthentication',
)}
```

→ open the views.py of Applications and the add the following code:

```
from django.shortcuts import render
from rest_framework...
```



```
from .models import Product, Review
from .serializers import ProductSerializer, ReviewSerializer
from rest_framework import ...
```

```
class ProductList(generics.RetrieveUpdateDestroyAPIView):
    queryset = product.objects.all()
    serializer_class = ProductSerializer
    Permission_classes = (IsAdminOrReadOnly, )

class ReviewDetail(generics.RetrieveUpdateDestroyAPIView):
    Serializer_class = ReviewSerializer
    Permission_classes = (IsAuthenticatedOrReadOnly,
                          IsOwnerOrReadOnly)
```

→ open the permissions.py file and the following code :

```
from rest_framework.permissions import BasePermission, SAFE_METHODS
class IsAdminOrReadOnly(BasePermission):
    def has_permission(self, request, view):
        if request.methods in SAFE_METHODS:
            return True

        else
            return request.user.is_staff

class IsOwnerOrReadOnly(BasePermission):

    def has_object_permission(self, request, view, obj):
        if request.method in SAFE_METHODS:
            return True

        return obj.created_by = request.user
```

→ open the models.py of Application and add the code:

```
from django.db import models
from django.contrib.auth.models import User
class Product(models.Model):
    pid = models.IntegerField(primary_key=True)
    pname = models.CharField(max_length=20)
    pcost = models.DecimalField(max_digits=10,
                                decimal_places=4)
    pmfd = models.DateField()
    pexpdt = models.DateField()
class Review(models.Model):
    product = models.ForeignKey(Product, on_delete=
              models.CASCADE, related_name='reviews')
    title = models.CharField(max_length=255)
    review = models.TextField()
    rating = models.IntegerField()
    created_by = models.ForeignKey(User, null=True,
                 on_delete=models.CASCADE)
```

→ Open the serializers.py of Application and add the following code:

```
from rest_framework import serializers
from .models import Product, Review
class ProductSerializer(serializers.ModelSerializer):
    class Meta:
        model = Product
        fields = ('pid', 'pname', 'pcost', 'pmfd', 'perpd')
class ReviewSerializer(serializers.ModelSerializer):
    created_by = serializers.ReadOnlyField(source='created_by.
                                            username'),
    class Meta:
        model = Review
        fields = ('id', 'title', 'review', 'ratings', 'created_by')
```

→ Open the urls.py of Application and add the following code:

```
from django.conf.urls import url
from . import views
app_name = 'restappReview'
urlpatterns = [
url(r'^products/(?P<pk>[0-9]+)$', views.ProductList.as_view()),
url(r'^products/(?P<product_id>[0-9]+)/reviews/(?P<review_id>[0-9]+
                                                    )/$',
    views.ReviewDetail.as_view()),
]
```

→ open the urls.py of the project and add the following code:

```
from django.conf.urls import url, include
from rest_framework.authtoken.views import obtain_auth_token

urlpatterns = [

    url(r'^api-token-auth/', obtain_auth_token),
    url(r'^restappReview/', include('restappReview.urls')),

]
```

RestAPI - Project - I

→ Create a Restful API for movies (Something similar to IMDB)

1. MySQL or SQLLite3 to store data
2. Use Django Frame Work for implementing the APIs

these need to be 2 level of access:

Admin = who can add, remove or edit movies

Users = who can just view the movies

→ There should also be a decent implementation to search for movies

→ Document your code well so that we can test the API With ease.

→ For your convenience I have attached some data that you can use to populate your database

→ Feel free to add features!

Sample output

```
[
  {
    "99popularity": 83.0,
    "director": "Victor Fleming",
    "genre": [
      "Adventure",
      "Family",
      "Fantasy",
      "Musical",
    ],
```

```
"imdb_score": 8.3,
"name": "the Wizard of Oz"
},
{
"qqpopularity": 88.0
"director": "George Lucas",
"genre": [
"Action",
"Adventure",
"Fantasy",
"Sci-Fi"
],
"imdb_score":8.8,
"name": "Stars Wars"
}
]
```

RestAPI Project - II

Asteroid - Neo Stats

→ Neo stands for Near Earth Objects. Nasa provides an open API and in this problem we will be using the Asteroid NeoWs API

→ We want to plot a line chart showing number of asteroids passing near earth each day for the given date range as well as find the nearest asteroid and the fastest asteroid

Data Source

Retrieve a list of Asteroids based on their closest approach date to Earth.

http://api.nasa.gov/api.html#neows-feed

Web Application

→ this should be a Single page Web Application. Feel free to use any reactive Js libraries like VueJs, AngularJs or anything you like. Use Twitter Bootstrap for the UI

→ As a user, I want to select start and end date so that I can view the Neo Stats for that date range.

→ provide a way (input) for the user to specify the start and end dates. Use a date picker for the same

→ Once the dates are selected, user will hit "Submit". On Submit, fetch the Neo Feed from NASA's Open API for the given date range and plot a graph showing total number of asteroids for each day of the given date range. Use a bar or line chart for the same.

→ Also show the following stats (deduced from the data you will recieve from Neo Feed)

1. Fastest Asteroid in km/h
2. Closest Asteroid
3. Average Size of the Asteroids in Kilometers

email : lokesh.rapala@gmail.com