Procedures

1.Creating Virtual Environment

```
python -m venv .venv
```

2.Activate the Script

```
    .venv\Scripts\activate
```

3.Install Packages

```
pip install djangorestframework

pip install mysqlclient   (MySQL DATABASE)
```

4.Create Project

```
django-admin startproject P1_BillingSystem .
```

5.Create Application

```
python manage.py startapp Inventory
```

6.Project Settings File Register the Application

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

7.Start The Server Once to avoid Errors.

```
python manage.py runserver
```

8. Project Settings File Set the Database you need

8.1 For SQLITE3 Database

```
In VScode Install sqliteviewer Extension.
It helps to view Sqlite Database.
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

8.2 For MySQL DATAbase (MySQL WorkBench)

pip install mysqlclient

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'P1_Billing',
        'USER':'root',
        'PASSWORD':'',
        'HOST':'localhost',
        'PORT':'3306'
    }
}
```

8.3 MySQL Database(Wamp Server)

pip install mysqlclient

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'P1_Billing',
        'USER':'root',
        'PASSWORD':'',

    }
}
```

9.Create The Database in Mysql Console or in Work Bench.

```
MySQL Commands

show databases;
create database P1_Billing;
show databases;
```

10.Then Create your Tables in Application

Inventory/models.py

```
from django.db import models

# Create your models here.
class Productdetails(models.Model):
    product_name = models.CharField(max_length=200,null=True)
    code         = models.CharField(max_length=200,null=True)
    price        = models.FloatField(default=0)
```

11.Migrate the Database in Terminal

python manage.py makemigrations

python manage.py migrate

PROJECT 1

- 1.Here the API CALLS are prepared using the Class Based.
- 2.Here MySQL Database are used(Wamp Server).
- 3.Here No Django Serializing is Done.

PROJECT -1

CODINGS

```
### Project urls.py

from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inventory/',   include('Inventory.urls')),
]
```

```
### Application urls.py

from django.urls import path
from .views import *

urlpatterns = [
    path('products/',          Ourproducts.as_view() ),
    path('products/<int:id>/', Ourproductsbyid.as_view() ),

]
```

```
### Application models.py

from django.db import models

# Create your models here.
class Myproducts(models.Model):
    product_name = models.CharField(max_length=200,null=True)
    code         = models.CharField(max_length=200,null=True)
    price        = models.FloatField(default=0)
```

```
### Application views.py

from rest_framework .views import APIView
from rest_framework .response import Response
from .models import *


class Ourproducts(APIView):
    def post(self,request):
        uservalues = request.data
        print(uservalues)
```

```
        new_product =
Myproducts(product_name=uservalues['product_name'],code=uservalues['code'],price=user
values['price'] )
        new_product.save()
        return Response("New Data Saved")

    def get(self,request):
        products = Myproducts.objects.all()
        fullproducts = []
        for each in products:
            singleproduct = {
                'product_name' : each.product_name,
                'product_code' : each.code,
                'product_price': each.price,
                'product_id'   : each.id,
            }
            fullproducts.append(singleproduct)

        return Response(fullproducts)



class Ourproductsbyid(APIView):
    def get(self,request,id):
        detail = Myproducts.objects.get(id=id)
        product = {
                    'product_name':detail.product_name,
                    'product_code':detail.code,
                    'product_price':detail.price,
                    'product_id': detail.id
        }
        return Response(product)

    def put(self,request,id):
        detail    = Myproducts.objects.get(id=id)
        uservalue = request.data
        detail.product_name = uservalue['name']
        detail.code         = uservalue['code']
        detail.price        = uservalue['price']
        detail.save()
        return Response('Data Updated Successfully ')

    def patch(self,request,id):
        detail    = Myproducts.objects.get(id=id)
        uservalue = request.data
        detail.product_name = uservalue['name']
        detail.code         = uservalue['code']
        detail.price        = uservalue['price']
        detail.save()
        return Response('Data Updated Successfully by Patch ')

    def delete(self,request,id):
        detail   = Myproducts.objects.get(id=id)
        detail.delete()
        return Response('Deleted Successfully'
```

API OUTPUT TABLE

No	Method	Input	Output

1	**POST**
	http://127.0.0.1:8000/inventory/products/
		<pre>{
		"product_name":"Orange",
		"code":"PRO1003",
		"price": 300
		}</pre>
			<pre>New Data Saved</pre>

2	Get
	http://127.0.0.1:8000/inventory/products/
			<pre>[
			{
			"product_name": "Orange",
			"product_code": "PRO1001",
			"product_price": 300.0,
			"product_id": 1
			},
			{
			"product_name": "Apple",
			"product_code": "PRO1002",
			"product_price": 200.0,
			"product_id": 2
			}
			]</pre>

3	**GET**
	http://127.0.0.1:8000/inventory/products/1/
			<pre>{
			"product_name": "Apple",
			"product_code": "PRO1002",
			"product_price": 200.0,
			"product_id": 2
			}</pre>

4	**PUT**
	http://127.0.0.1:8000/inventory/products/1/
		<pre>{
		"name":"Strawberry",
		"code":"PRO1003",
		"price": 300
		}</pre>
			<pre>"Data Updated Successfully "</pre>

5	**PATCH**
	http://127.0.0.1:8000/inventory/products/2/
		<pre>{
		"name":"Sappoto",
		"code":"PRO1004",
		"price": 400
		}</pre>
			<pre>"Data Updated Successfully by Patch "</pre>

6	**DELETE** http://127.0.0.1:8000/inventory/products/3/		<pre>"Deleted Successfully"</pre>

PROJECT 2

CODINGS

```
### Project urls.py
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inventory/', include('Inventory.urls')),
]
```

```
### Application urls.py
from django.contrib import admin
from django.urls import path
from .views import *

urlpatterns = [
    path('products1s/', Ourproducts1.as_view()),
    path('products2s/', Ourproducts2.as_view()),
    path('products/<int:id>/', OurproductsbyID.as_view()),
]
```

```
### Application Views.py

from rest_framework .views import APIView
from rest_framework .response import Response
from .models import *
from .serializers import *

class Ourproducts1(APIView):

   def get(self,request):
        full_products = Myproducts.objects.all()
        serialized_products =
Products_Serializer1(full_products,many=True).data       ## Many Records
        return Response(serialized_products)


class Ourproducts2(APIView):
    def get(self,request):
        full_products = Myproducts.objects.all()
        serialized_products =
Products_Serializer2(full_products,many=True).data       ## Many Records
        return Response(serialized_products)


class OurproductsbyID(APIView):
    def get(self,request,id):
```

```
        req_product = Myproducts.objects.get(id=id)
        serialized_products = Products_Serializer1(req_product).data
        return Response(serialized_products)

    def put(self,request,id):
        detail = Myproducts.objects.get(id=id)
        uservalues = request.data

        detail.product_name = uservalues['name']
        detail.code = uservalues['code']
        detail.price = uservalues['price']
        detail.save()
        return Response(f"Updated Product as  {uservalues['name']}")

    def delete(self,request,id):
        detail = Myproducts.objects.get(id=id)
        detail.delete()
        return Response(f"Deleted ID {id}")
```

```
### APP serializer.py
from rest_framework import serializers
from .models import *

class Products_Serializer1(serializers.ModelSerializer):

    class Meta:
        model = Myproducts
        fields = '__all__'    #For all fields in model


class Products_Serializer2(serializers.ModelSerializer):

    class Meta:
        model = Myproducts
        fields = ['product_name']    #For single Field
```

```
### App Models.py
from django.db import models

# Create your models here.
class Myproducts(models.Model):
    product_name = models.CharField(max_length=200,null=True)
    code         = models.CharField(max_length=200,null=True)
    price        = models.FloatField(default=0)
```

PROJECT 2 - API OUTPUT TABLE

1	Method/Url	Input	Output

	Get
	http://127.0.0.1:8000/inventory/products1s/
			[
			{
			"id": 1,
			"product_name": "Apple",
			"code": "PRO1003",
			"price": 300.0
			},
			{
			"id": 2,
			"product_name": "Orange",
			"code": "PRO1002",
			"price": 1800.0
			}
			]

2	GET
	http://127.0.0.1:8000/inventory/products2s/
			[
			{
			"product_name": "Apple"
			},
			{
			"product_name": "Orange"
			}
			]

3	GET
	http://127.0.0.1:8000/inventory/products/1/
			{
			"id": 1,
			"product_name": "Apple",
			"code": "PRO1003",
			"price": 300.0
			}

PROJECT 3

```python
### Project urls.py
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('inventory/', include('Inventory.urls')),
]
```

```python
### Application Urls.py
from django.contrib import admin
from django.urls import path
```

```python
from .views import *

urlpatterns = [
    ### Category with Serializer
    path('category/',           OurCategory.as_view()),
    path('category/<int:id>/',  OurCategory.as_view()),

    ## Products with serializer
    path('ser_products/',           Ourproducts_serial.as_view()),
    path('ser_products/<int:id>/',  Ourproducts_serialBYID.as_view()),

     ## Products without serializer
    path('man_product/',            Ourproducts_manual.as_view()),
    path('man_products/<int:id>/',  Ourproducts_manualBYID.as_view()),

]
```

```python
### Application views.py
from rest_framework .views import APIView
from rest_framework . response import Response
from .models import *
from .serializers import *

### Category Foreignkey Connection
class OurCategory(APIView):
    def post(self,request):
        uservalues = request.data
        ProductCategory(category_name=uservalues['category_name']).save()
        return Response ("New Item Saved in Category")

    def get(self,request):
        categoryitems =ProductCategory.objects.all()
        serialized_category =  Category_Serializer(categoryitems,many=True).data
        return Response(serialized_category)

    def get(self,request,id):
        categoryitem =ProductCategory.objects.get(id=id)
        serialized_category =  Category_Serializer(categoryitem).data
        return Response(serialized_category)



### Products Serialized Method

class Ourproducts_serial(APIView):
    def post(self,request):
        new_product =  Product_serializer(data=request.data)
        if new_product.is_valid():
            return Response("New Product using Serializer")
        else:
            return Response(new_product.errors)

    def get(self,request):
```

```
        viewproducts = Myproducts.objects.all()
        serialized_products =  Product_serializer(viewproducts,many=True).data
        return Response(serialized_products)


class Ourproducts_serialBYID(APIView):
    def get(self,request,id):
        viewproducts = Myproducts.objects.get(id=id)
        serialized_products =  Product_serializer(viewproducts).data
        return Response(serialized_products)

    def put(self,request,id):
        viewproducts = Myproducts.objects.get(id=id)
        serialized_data = Product_serializer(viewproducts, data=request.data)
        if serialized_data.is_valid():
            serialized_data.save()
            return Response('Updated')
        return Response('Error in Updating')

    def delete(self,request,id):
        product = Myproducts.objects.get(id=id)
        product.delete()
        return Response("Deleted")



### Products Manual Method

class Ourproducts_manual(APIView):
    def post(self,request):
        uservalues = request.data
        new_product =
Myproducts(product_name=uservalues['product_name'],code=uservalues['code'],price=user
values['price'] ,category_refrences_id=uservalues['category'])
        new_product.save()
        return Response("New Product  Created by Manual Method")


    def get(self,request):
        products = Myproducts.objects.all()
        fullproducts = []
        for each in products:
            singleproduct = {  'product_name' : each.product_name, 'product_code' :
each.code, 'product_price':     each.price,'product_id'    : each.id,
'category_id':each.category_refrences_id }
            fullproducts.append(singleproduct)
        return Response(fullproducts)

class Ourproducts_manualBYID(APIView):
    ### GETBYID
    def get(self,request,id):
        detail = Myproducts.objects.get(id=id)
        product = {  'product_name':detail.product_name,  'product_code':detail.code,
'product_price':detail.price,
```

```
                    'product_id': detail.id ,
'category_id':detail.category_refrences_id}
        return Response(product)

    def put(self,request,id):
        detail    = Myproducts.objects.get(id=id)
        uservalue = request.data
        detail.product_name = uservalue['product_name']
        detail.code         = uservalue['code']
        detail.price        = uservalue['price']
        detail.category_refrences_id        = uservalue['category']
        detail.save()
        return Response('Data Updated Successfully ')
```

Models.py

```
## Application models.py
from django.db import models

### Parent Model so top
class ProductCategory(models.Model):
    category_name = models.CharField(max_length=200,null=True)

    def __str__(self):
        return self.category_name



### It deponds on Category(Child Model)
class Myproducts(models.Model):
    category_refrences =
models.ForeignKey(ProductCategory,null=True,on_delete=models.SET_NULL)
    product_name = models.CharField(max_length=200,null=True)
    code         = models.CharField(max_length=200,null=True)
    price        = models.FloatField(default=0)

    def __str__(self):
        return self.product_name
```

serializer.py

```
### Application Serializer.py
from rest_framework import serializers
from .models import *


class Category_Serializer(serializers.ModelSerializer):

    class Meta:
        model = ProductCategory
        fields = '__all__'    #For all fields in model
```

```python
class Product_serializer(serializers.ModelSerializer):
    class Meta:
        model = Myproducts
        fields = '__all__'
```

No	URL/METHODS	Input	Output

1	POST
	http://127.0.0.1:8000/inventory/category/
		{
		"category_name":"vegetables"
		}
			"New Item Saved in Category"

2	Get
	http://127.0.0.1:8000/inventory/category/
			[
			{
			"id": 1,
			"category_name": "flowers"
			},
			{
			"id": 2,
			"category_name": "fruits"
			},
			{
			"id": 3,
			"category_name": "vegetables"
			}
			]

3	Get http://127.0.0.1:8000/inventory/category/2/		{ "id": 2, "category_name": "fruits" }
	Serialized METHOD		

1	POST
	http://127.0.0.1:8000/inventory/ser_products/
		{
		"product_name":"Mango",
		"code":"PRO1002",
		"price":2360,
		"category":3
		}
			"New Product using Serializer"

2	GET http://127.0.0.1:8000/inventory/ser_products/		[{ "id": 1, "product_name": "Onion", "code": "PRO1001", "price": 100.0, "category_refrences": 3 }, {

			"id": 2, "product_name": "Beans", "code": "PRO1003", "price": 160.0, "category_refrences": 3 }]

3	GET
	http://127.0.0.1:8000/inventory/ser_products/2/
			{
			"id": 2,
			"product_name": "Beans",
			"code": "PRO1003",
			"price": 160.0,
			"category_refrences": 3
			}

4	PUT
	http://127.0.0.1:8000/inventory/ser_products/3/
		{
		"id": 3,
		"product_name": "BMW",
		"code": "PRO1003",
		"price": 123.65,
		"category_refrences": 1
		}
			Updated

5	DELETE http://127.0.0.1:8000/inventory/ser_products/2/		"Deleted"
	Manual		

1	POST
	http://127.0.0.1:8000/inventory/man_products/
		{
		"product_name":"Mango",
		"code":"PRO1002",
		"price":2360,
		"category":3
		}
			Created

2	GET http://127.0.0.1:8000/inventory/man_product/		[{ "product_name": "Onion", "product_code": "PRO1001", "product_price": 100.0, "product_id": 1, "category_id": 3 }, { "product_name": "Audi", "product_code": "PRO10023", "product_price": 123.65, "product_id": 3, "category_id": 1 }, { "product_name": "Berry",

			"product_code": "PRO1001", "product_price": 100.0, "product_id": 4, "category_id": 3 }]

3	GET http://127.0.0.1:8000/inventory/man_products/3/		{
			"product_name": "Audi",
			"product_code": "PRO10023",
			"product_price": 123.65,
			"product_id": 3,
			"category_id": 1
			}

4	PUT http://127.0.0.1:8000/inventory/man_products/3/	{
		"id": 3,
		"product_name": "Audi",
		"code": "PRO10023",
		"price": 123.65,
		"category": 1
		}
			Updated