* In REST API start for Project same as previously done Django Project commands and Libraries all are same but some commands are added in that
* Like Serializers
* Make Folder for Project for store the files
* Open sublime Text and access the same
* Open CMD PROMPT and access the same folder
* Create Virtual Environment  
  **python -m venv myvenv**
* Access the virtual Environment  
  **myvenv\Scripts\activate**
* Install Django in Project  
  **pip install Django**
* Create Project  
  **django-admin startproject mysite .**
* Create App for our Project  
  **Django-admin startapp myapp**
* Save into the Database  
  **python manage.py migrate**
* Go to settings.py and register our app in INSTALLED APP  
  **INSTALLED\_APPS = [**

**'myapp',**

**]**

* Install DjangoRestFramework  
  **pip install djangorestframework**
* Now register djangoRestFramework  
  **rest\_framework** /// in settings.py add in INSTALLED APP
* Create class  
  **class Book(models.Model):**

**title = models.CharField(max\_length = 100,blank = True)**

**author = models.CharField(max\_length = 100,blank = True)**

**isbn = models.CharField(max\_length = 100,blank = True)**

**publisher = models.CharField(max\_length = 100,blank = True)**

**def \_\_str\_\_(self):**

**return self.title**

* **python manage.py makemigrations**
* **python manage.py migrate**
* create sterilizers file in myapp folder  
  **serializers.py**Sure! Here's a simplified version in very easy language:
  + Turn Data into JSON : Serializers change complex data (like database records) into simple formats (like JSON) that can be sent over the internet.
  + Read Incoming Data : They also read incoming JSON data and turn it back into complex data types.
  + Check Data\*\*: They make sure the data is correct and follows the rules.
  + Easy with Models : If you're using Django models, ModelSerializer makes it easy by automatically using the model's fields.
  + Custom Rules : You can add your own rules for how data should be handled.

This way, serializers help send and receive data in a format that both your Django app and web clients can understand.

* Paste below code in serializers.py  
  **from rest\_framework import serializers**

**from .models import Book**

**class BookSerializer(serializers.ModelSerializer):**

**class Meta:**

**model = Book**

**fields = ('id','title','author','isbn','publisher')**

* Go to views.py and pase below code  
  **from django.shortcuts import render**

**from rest\_framework import generics**

**from .models import Book**

**from .serializers import BookSerializer**

**class BookList(generics.ListCreateAPIView):**

**queryset = Book.objects.all()**

**serializer\_class = BookSerializer**

**class BookDetail(generics.RetrieveUpdateDestroyAPIView):**

**queryset = Book**

**serializer\_class = BookSerializer**

* Go to **mysite > urls.py** and paste below code  
  **from django.contrib import admin**

**from django.urls import path**

**from myapp.views import BookList,BookDetail**

**urlpatterns = [**

**path('api/books',BookList.as\_view()),**

**path('api/books/<int:pk>',BookDetail.as\_view()),**

**path('admin/',admin.site.urls),**

**]**

* After that run the server to check whether it is work or not **python manage.py runserver**
* Now open the url in the browser to check our project **http://localhost:8000/api/books**

**How to TEST our REST API Project using POSTMAN**

* **Create New Collection and Rename it**
* Create **ADD REQUEST** and rename it **Get All DATA  
  Method = GET   
  URL = localhost:8000/api/books**
* Create Another Request and named **= Insert Data**Method = POST  
  URL = [**http://localhost:8000/api/books**](http://localhost:8000/api/books)

Go to BODY TAG and then go to RAW Tabe and type field you want as per our class{

    "title":"Core Java",

    "author":"Akshay",

    "isbn":"78945612",

    "publisher":"Tops"

}

* Create UPDATE REQUEST   
  Method = Get ///Access the data you want to update  
  URL = [**http://localhost:8000/api/books/12**](http://localhost:8000/api/books/12) **Method = PUT  
  URL =** [**http://localhost:8000/api/books/12**](http://localhost:8000/api/books/12)Change according to your requirement and click on SEND Btn
* Create New Request name = **DELETE DATA**

**Method = GET //**Access the Data by Get Method **URL =** [**http://localhost:8000/api/books/12**](http://localhost:8000/api/books/12) **Method = DELETE //** After Access that data and delete the same using DELETE Method **URL =** [**http://localhost:8000/api/books/12**](http://localhost:8000/api/books/12)