Django REST Framework

What is an API?

- API stands for Application Programming Interface
- It acts as a two-way communication bridge between frontend and backend



Frontend & Backend Communication

What is REST API?

- REST stands for Representational State Transfer
- It organizes how web applications talk to each other, separating what the user sees (frontend) from what runs behind the scenes (backend).

Core Principles of REST

Stateless:

The server doesn't store any information about the client between requests.

Client-Server Architecture:

The app (client) asks for things (data) and the server does what's requested (sends data or makes changes).

Core Principles of REST

Standardized Interface:

REST APIs rely on a set of standard methods (GET, POST, PUT, DELETE) for interacting with resources.

GET - retrieving the resource

PUT - updating the resource

POST - creating the resource

DELETE - deleting the resource

Core Principles of REST

Easy-to-Read Data:

REST APIs returns the response in a standardized easy to read formats, typically JSON or XML formats.

RESTful Operations

GET, POST, PUT, DELETE

	URI
GET	/students
POST	/students
PUT	/student/1
DELETE	/student/1

Virtual Environment

Web Application Endpoints

Users can directly access it from the web browsers

http://127.0.0.1:8000/students/

API Endpoints

Returns the data to integrate into the frontend.

http://127.0.0.1:8000/api/v1/students/

Serializers

Serialization is the process of converting the model object or a queryset into JSON format or any other formats that the client requires.

Class Based Views

Class-based views provide more structured and organized way to handle requests using Object-oriented principles.

get()	→	Get the students
post()	→	Create a student
put()	→	Update a student
delete()	→	Delete a student

Mixins

Mixins are reusable code classes in object-oriented programming that provide specific functionalities.

In Django REST Framework, mixins are used to add common functionality to views.

Create

Read

Update

Delete

Mixins

class Employees(mixins, generics.GenericAPIView)

ListModelMixin	list()	generics.GenericAPIView
CreateModelMixin	create()	get()
RetriewModelMixin	retrieve()	post()
UpdateModelMixin	update()	put()
DestroyModelMixin	destroy()	delete()

```
{
  "id": 1,
  "name": "John Doe",
  ”designation":  ”Software Engineer"
}
```

Mixins

ListModelMixin

CreateModelMixin

RetriewModelMixin

UpdateModelMixin

DestroyModelMixin

```
class Employees(mixins.ListModelMixin, generics.GenericAPIView):

    def get(self, request):
        return self.list(request)
```

```
class Employees(mixins.ListModelMixin, mixins.CreateModelMixin, generics.GenericAPIView):

    def get(self, request):
        return self.list(request)

    def post(self, request):
        return self.create(request)
```

Generics

ListAPIView	⇒	For listing the objects
CreateAPIView	⇒	For creating the objects
RetrieveAPIView	⇒	For retrieving a single object using pk
UpdateAPIView	⇒	For updating a single object using pk
DestroyAPIView	⇒	For deleting an object using pk

ListCreateAPIView	⇒	For listing and creating objects
RetrieveUpdateAPIView	⇒	For retrieving & updating objects using pk
RetrieveUpdateDestroyAPIView	⇒	For retrieving, updating & deleting objects using pk

Viewsets

viewsets.ViewSet

- list()
- create()
- retrieve()
- update()
- delete()

viewsets.ModelViewSet

Takes only queryset and serializer_class and automatically provides both pk based and non-pk based operations.

Pagination

PageNumberPagination

/employees/?page=10

page_size = 10

Takes a page_size parameter and returns the response accordingly.

LimitOffsetPagination

/employees/?limit=10&offset=0

Limit: This parameter controls how many items you want to see in a single page.

Offset: This parameter tells the API where to start fetching the items from.

If offset=0, you get the first 10 emps (items 1-10).

If offset=10, you get the next 10 emps (items 11-20).

If offset=90, you get the last 10 emps (items 91-100).

/employees/?limit=10&offset=0

/employees/?limit=10&offset=10

/employees/?limit=10&offset=90

Filtering

Use Case 1: Filter Employees by Designation

Criteria: Retrieve all employees with the designation "Software Engineer."

Assignment: Use Case 2: Filter Employees by Name

Criteria: Retrieve all employees whose names contain the word "John."

Assignment: Use Case 3: Filter Employees by ID Range

Criteria: Retrieve all employees with IDs between ”EMP002" and ”EMP006."