DRF - Task - I

What is Django REST Framework?

Django REST Framework (DRF) serves as a robust toolkit for crafting Web APIs within the Django framework. With a comprehensive set of tools and functionalities, DRF streamlines the development of scalable and resilient APIs in Django.

Highlighted below are some prominent features and components of Django REST Framework:

Serialization: DRF boasts a flexible serialization system, facilitating the effortless conversion of Django model instances and querysets into various content types like JSON. Additionally, serializers handle deserialization, enabling the conversion of data back into intricate Python objects.

Views: DRF offers an array of pre-built class-based views tailored for common API patterns, such as CRUD operations (Create, Retrieve, Update, Delete), list and detail views, and more. These views are adaptable and equipped with hooks for incorporating custom behaviours.

Authentication and Permissions: DRF provides an assortment of authentication and permission classes to fortify APIs. It supports diverse authentication methods including token-based authentication, session authentication, OAuth, etc. Permissions can be fine-tuned to regulate access to resources based on user roles and permissions.

Pagination: DRF facilitates seamless pagination, enabling APIs to furnish paginated results for extensive datasets. Pagination classes can be personalized to manage pagination behaviours effectively.

Throttling: DRF integrates throttling classes to enforce rate limiting on API requests. Throttling can be configured based on various factors like request count per user, per IP address, or per resource.

Browsable API: DRF's browsable API furnishes a web-based interface for exploring and interacting with APIs directly from a browser. It empowers developers to effortlessly inspect API endpoints, submit requests, and scrutinize responses in a user-friendly manner.

Serialization Formats: DRF supports an array of serialization formats, encompassing JSON, XML, YAML, and more. Additionally, it extends support for custom serialization formats, offering flexibility as per project requirements.

In essence, Django REST Framework stands as a cornerstone within the Django community for API development, attributable to its rich feature set, user-friendly nature, and extensive documentation. It streamlines the API development process in Django, while delivering the adaptability and scalability demanded by complex projects.

What is restful API ?

RESTful API (Representational State Transfer API) is an architectural style for designing networked applications. It is based on the principles of REST, which was originally outlined by Roy Fielding in his doctoral dissertation in 2000.
Key principles of RESTful APIs include:

1. **Client-Server Architecture:** The client and server are separate entities that communicate over a network. The client makes requests to the server, and the server processes those requests and returns responses.
2. **Statelessness:** Each request from a client to the server must contain all the information necessary for the server to understand and process the request. The server does not maintain any client state between requests. This simplifies the server implementation and improves scalability.
3. **Uniform Interface:** A uniform interface between clients and servers promotes a shared understanding of how to interact with the API. This includes using standard HTTP methods (GET, POST, PUT, DELETE) for performing CRUD (Create, Read, Update, Delete) operations, and using resource identifiers (URLs) to uniquely identify resources.
4. **Resource-Based:** Resources are the key abstraction in RESTful APIs. Every resource is uniquely identified by a URL, and clients interact with these resources using standard HTTP methods. Resources can represent entities such as users, articles, products, etc.
5. **Representation:** Resources are represented in a format that can be easily consumed by clients, such as JSON or XML. Clients can request different representations of a resource using content negotiation.
6. **State Transfer:** The state of a resource is transferred between the client and server through representations. Clients can retrieve, create, update, or delete resources by transferring representations of those resources over HTTP.

Overall, RESTful APIs provide a flexible and scalable approach to building distributed systems by leveraging the existing infrastructure of the World Wide Web. They promote simplicity, reliability, and interoperability, making them a popular choice for building modern web and mobile applications.

Create a model role and link with the user model?

settings.py | models.py | serializers.py | views.py | urls.py ...\app | admin.py | urls.py ...\project

project > app > serializers.py > ...

```python
from rest_framework import serializers
from django.contrib.auth.models import User
from .models import Role

class UserSerializer(serializers.ModelSerializer):
    password = serializers.CharField(write_only=True)

    class Meta:
        model = User
        fields = ('username', 'password', 'email', 'first_name', 'last_name')

    def create(self, validated_data):
        user = User.objects.create_user(**validated_data)
        return user

class RoleSerializer(serializers.ModelSerializer):
    class Meta:
        model = Role
        fields = '__all__'

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[16/Apr/2024 15:30:34] "GET /static/rest_framework/js/csrf.js HTTP/1.1" 304 0
[16/Apr/2024 15:30:34] "GET /static/rest_framework/js/prettify-min.js HTTP/1.1" 304 0
[16/Apr/2024 15:30:34] "GET /static/rest_framework/js/bootstrap.min.js HTTP/1.1" 304 0
[16/Apr/2024 15:30:34] "GET /static/rest_framework/js/load-ajax-form.js HTTP/1.1" 304 0
[16/Apr/2024 15:30:34] "GET /static/rest_framework/js/default.js HTTP/1.1" 304 0
[16/Apr/2024 15:30:34] "GET /static/rest_framework/img/grid.png HTTP/1.1" 304 0
Method Not Allowed: /app/register/
[16/Apr/2024 15:30:37] "GET /app/register/ HTTP/1.1" 405 8510
```

settings.py | models.py | serializers.py | views.py | urls.py ...\app | admin.py | urls.py ...\project

project > app > models.py > ...

```python
from django.contrib.auth.models import User
from django.db import models

class Role(models.Model):
    name = models.CharField(max_length=100)
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.name

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python

```
[16/Apr/2024 15:30:34] "GET /static/rest_framework/js/csrf.js HTTP/1.1" 304 0
[16/Apr/2024 15:30:34] "GET /static/rest_framework/js/prettify-min.js HTTP/1.1" 304 0
[16/Apr/2024 15:30:34] "GET /static/rest_framework/js/bootstrap.min.js HTTP/1.1" 304 0
[16/Apr/2024 15:30:34] "GET /static/rest_framework/js/load-ajax-form.js HTTP/1.1" 304 0
[16/Apr/2024 15:30:34] "GET /static/rest_framework/js/default.js HTTP/1.1" 304 0
[16/Apr/2024 15:30:34] "GET /static/rest_framework/img/grid.png HTTP/1.1" 304 0
Method Not Allowed: /app/register/
```

Create API to register user?