Answer the following

1. What is REST API? State different REST API request Method.

   R**EST API (Representational State Transfer Application Programming Interface)** is a standardized architecture for building web services. It uses HTTP methods to allow communication between client and server, often in JSON format.

   A **RESTful API** follows specific constraints such as:

- Statelessness: Each request contains all the information needed.
- Uniform Interface: Consistent structure and access patterns.
- Client-Server Separation: Backend and frontend are independent.
- Cacheable: Responses can be cached to improve performance.

HTTP Method	Purpose	Description
GET	Retrieve data	Fetches data from the server (e.g., a list of users).
POST	Create new data	Sends data to the server to create a resource.
PUT	Update existing data (full)	Replaces an existing resource completely.
PATCH	Update existing data (partial)	Modifies part of a resource.
DELETE	Delete data	Removes a resource from the server.

2. What is Django Rest Framework?

   Django REST Framework (DRF) is a powerful and flexible toolkit built on top of Django to create **RESTful APIs**.

   Key features:

- Serialization of data (models → JSON and vice versa)
- Authentication and Permissions
- Browsable API interface
- ViewSets and Routers for simplified routing
- Generic views and class-based views for CRUD operations

  It helps you expose Django models and logic via an API quickly and securely.

3. What is Serializers and De-serializers in Django Rest Framework?

Serializers

- Convert **complex data types** like Django **models** or Python **objects** into **JSON** (or other content types) for API responses.

- Example:

```
class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username', 'email']
```

Deserializers

- Convert **JSON or other formats** received from the API (e.g., in a POST request) back into Django **model instances** or **Python objects**.
- They also handle **validation** and **type conversion** during this process.

Both serialization and deserialization are handled by the same Serializer class depending on context.