REST FRAMEWORK:

1. What is an API?

An API (Application Programming Interface) is a way for two software applications to talk to each other.
It works like a **middleman** that sends requests and returns responses.

In simple words:
One app asks for data → API sends the data back.

Types of APIs

REST API

- Most commonly used
- Uses HTTP methods like GET, POST, PUT, DELETE
- Data is shared in JSON format
- Fast, simple, and lightweight

SOAP API

- Uses XML format
- Very strict rules
- More secure
- Mostly used in banking and enterprise systems

Why APIs are important?

- Connect frontend and backend
- Save development time
- Allow third-party integrations
- Make applications scalable and flexible

---

2. Requirements for Web Development Projects

Understanding project requirements

This is the first and most important step.
Developers must clearly understand:

- What the user wants
- Required features
- Target users
- Design expectations
- Performance needs

Good planning reduces mistakes and saves time and cost.

Setting up environment

Before coding, developers must:

- Install code editor
- Setup database
- Install frameworks and libraries

A proper setup helps in smooth development and testing.

---

3. What is Serialization?

Serialization means converting complex data (objects or querysets) into simple formats like JSON or XML.

Serialization in Django

- Django QuerySets cannot be sent directly
- Serializers convert them into JSON
- Django REST Framework provides powerful serializers
- Serializers also validate data before saving

---

4. HTTP Methods

GET

- Used to get data
- Does not change data

POST

- Used to create new data
- Example: user registration

PUT

- Used to update existing data completely

DELETE

- Used to remove data

Requests & Responses in DRF

- DRF views receive requests
- Responses are returned using `Response`
- Data is automatically converted into JSON
- HTTP status codes are handled easily

---

5. Views in DRF

Function-Based Views (FBVs)

- Written as simple Python functions
- Use `@api_view` decorator
- Easy to understand
- Best for small projects

Class-Based Views (CBVs)

- Written using Python classes
- Methods for GET, POST, PUT, DELETE
- Better structure and reusability
- Best for large projects

Difference

FBVs are simple but repetitive.
CBVs are clean, reusable, and scalable.

---

6. URLs and Views in Django

- URLs are defined in `urls.py`
- Each URL is linked to a view
- When a request matches a URL, the view runs
- This keeps the project organized

---

7. Pagination in APIs

Pagination means showing data in parts instead of all at once.

Why pagination?

- Improves performance

- Reduces server load
- Better user experience
- Useful for large datasets

DRF provides built-in pagination support.

---

8. Django Settings Configuration

Database settings

- Database name
- Username
- Password
- Host and port

This allows Django to store and fetch data.

Static files

- CSS, JavaScript, images
- Proper settings ensure correct loading

API keys

- Used for third-party services
- Stored securely in settings or environment variables

---

9. Setting Up Django REST Framework Project

Steps:

1. Install Django and DRF
2. Create project and app
3. Add DRF to settings
4. Create models
5. Create serializers
6. Write views
7. Configure URLs

This structure helps build clean and scalable APIs.

---

10. Social Authentication & Messaging

Social Login

- Login using Google or Facebook
- Uses OAuth
- Improves user experience
- More secure

Emails & OTPs

- Twilio → SMS and OTP
- SendGrid → Emails
- Used for verification and notifications
- Reliable and fast delivery

---

11. REST Principles

Statelessness

- Server does not store client session
- Every request has full information

Resource-based URLs

- URLs represent resources
- Example: `/users/`, `/products/`

HTTP Methods for CRUD

- GET → Read
- POST → Create
- PUT → Update
- DELETE → Delete

---

12. What is CRUD?

CRUD stands for:

- Create
- Read
- Update
- Delete

These are basic database operations used in all backend systems.

13. Authentication vs Authorization

Authentication

- Verifies user identity
- Example: login

Authorization

- Checks what the user is allowed to do
- Based on roles and permissions

Token Authentication in DRF

- Token generated after login
- Token sent with every request
- Secure and stateless
- Ideal for APIs

14. OpenWeatherMap API

- Provides real-time weather data
- Temperature, humidity, wind, forecast
- Requires API key
- Returns data in JSON format

---

15. Google Maps Geocoding API

- Converts address into latitude & longitude
- Used for maps and distance calculation
- Requires Google API key
- Returns JSON response

16. GitHub API

- Manage repositories
- Handle pull requests
- Create and update issues
- Useful for automation and integration

---

17. Twitter API

- Fetch tweets

- Post tweets
- Get user data
- Used for analytics and automation

---

18. REST Countries API

- Provides country details
- Capital, population, currency, language
- JSON response
- Useful for dashboards and learning apps

---

19. Email APIs (SendGrid / Mailchimp)

- Send automated emails
- Password reset, notifications
- High delivery rate
- Email analytics support

---

20. Twilio API

- Send SMS and OTP
- Used for authentication
- Fast and secure
- Supports global delivery

---

21. Payment Gateways (PayPal, Stripe)

- Process online payments
- Secure transactions
- Support cards and wallets
- Handle refunds and subscriptions

---

22. Google Maps API

- Display interactive maps
- Calculate distance and time
- Used for navigation and tracking

- Real-time location features