**Module 10) Rest Framework**

**1. What is an API (Application Programming Interface)?**

An API is a set of rules that allows different software applications to communicate with each other. It enables access to functionalities or data without exposing internal code.

**2. Types of APIs: REST, SOAP**

* **REST**: Lightweight, uses HTTP methods, returns data in JSON/XML, stateless.
* **SOAP**: Protocol-based, strict standards, uses XML, more secure for enterprise-level apps.

**3. Why are APIs important in web development?**

APIs enable communication between frontend and backend, integration with third-party services, and allow modular development with reusable components.

**4. Understanding project requirements**

Understanding the goal, user needs, features, and technical scope is essential for planning and implementing a successful web project.

**5. Setting up the environment and installing necessary packages**

Set up a virtual environment and install required packages using pip. Use requirements.txt to manage dependencies for consistency.

**6. What is Serialization?**

Serialization is the process of converting Django model instances or querysets into JSON or other formats that can be sent via APIs.

**7. Converting Django QuerySets to JSON**

Django REST Framework serializers automatically convert model data into JSON using the .data property of serialized objects.

**8. Using serializers in Django REST Framework (DRF)**

Serializers define the structure of API responses. They are created using serializers.ModelSerializer and used in views to transform data.

**9. HTTP request methods (GET, POST, PUT, DELETE)**

* **GET**: Retrieve data
* **POST**: Create new resource
* **PUT**: Update resource
* **DELETE**: Delete resource

**10. Sending and receiving responses in DRF**

DRF uses Response() to return data and status codes. It automatically handles JSON conversion and HTTP standards.

**11. Understanding views in DRF: Function-based views vs Class-based views**

* **Function-based**: Simple, good for small APIs
* **Class-based**: More structured, reusable, scalable for complex logic using APIView or GenericAPIView

**12. Defining URLs and linking them to views**

URLs are mapped to views using path() or re\_path() in urls.py. Each endpoint corresponds to a view handling the request.

**13. Adding pagination to APIs to handle large data sets**

Pagination breaks large results into smaller pages using PageNumberPagination or other classes in DRF. It improves performance and readability.

**14. Configuring Django settings for database, static files, and API keys**

Settings are defined in settings.py for database (e.g. SQLite/MySQL), static files, and third-party API keys like Twilio or SendGrid.

**15. Setting up a Django REST Framework project**

Start with django-admin startproject, create an app with startapp, install DRF, and configure settings to build API-ready projects.

**16. Implementing social authentication (e.g., Google, Facebook) in Django**

Use django-allauth and socialaccount.providers.google to integrate Google login with OAuth2 flow and user account creation.

**17. Sending emails and OTPs using third-party APIs like Twilio, SendGrid**

Twilio sends SMS/OTPs, SendGrid handles emails. Use respective SDKs and API keys to send messages from Django apps.

**18. REST principles: statelessness, resource-based URLs, and using HTTP methods for CRUD operations**

REST APIs are stateless, use nouns in URLs (e.g., /doctors/1/), and rely on HTTP methods (GET, POST, PUT, DELETE) for actions.

**19. What is CRUD, and why is it fundamental to backend development?**

CRUD stands for Create, Read, Update, Delete. It’s the basic operations needed to manage data in any database-backed application.

**20. Difference between authentication and authorization**

* **Authentication**: Verifies identity (e.g., login)
* **Authorization**: Grants permissions to access certain resources or perform actions

**21. Implementing authentication using Django REST Framework’s token-based system**

DRF uses TokenAuthentication where each user gets a token after login. Include the token in headers to access protected views.

**22. Introduction to OpenWeatherMap API and how to retrieve weather data**

OpenWeatherMap API provides weather data for cities. Use HTTP GET requests with API key and city name to fetch weather info.

**23. Using Google Maps Geocoding API to convert addresses into coordinates**

Send address to Google Maps API using HTTP request and receive latitude & longitude as JSON response using API key.

**24. Introduction to GitHub API and how to interact with repositories, pull requests, and issues**

GitHub API allows access to user data, repos, commits, and pull requests using endpoints and personal access tokens.

**25. Using Twitter API to fetch and post tweets, and retrieve user data**

Twitter API (via X API) allows applications to read tweets, post updates, and access profile data with OAuth credentials.

**26. Introduction to REST Countries API and how to retrieve country-specific data**

This public API gives country details like name, population, currency, and language based on country name/code.

**27. Using email sending APIs like SendGrid and Mailchimp to send transactional emails**

SendGrid and Mailchimp offer APIs to send automated, bulk, or transactional emails using templates and verified sender.

**28. Introduction to Twilio API for sending SMS and OTPs**

Twilio API enables sending OTPs or SMS through phone numbers using verified account credentials and Python SDK.

**29. Introduction to integrating payment gateways like PayPal and Stripe**

Stripe and PayPal provide secure APIs to process payments. You can charge customers, store cards, and manage subscriptions via backend.

**30. Using Google Maps API to display maps and calculate distances between locations**

Embed maps or calculate distances by sending location data to Google Maps API. Results return as coordinates or route metrics.