**1. Explain what a RESTful API is and its importance in mobile applications.**

A RESTful API (Representational State Transfer Application Programming Interface) is a software architectural style that allows systems to communicate with each other over the internet using standard HTTP methods. RESTful APIs are based on client-server architecture and use stateless communication, meaning that each request from a client (such as a mobile app) to the server must contain all the information the server needs to fulfill that request.

RESTful APIs typically return data in JSON (JavaScript Object Notation) or XML format, though JSON is the most commonly used due to its lightweight structure and compatibility with most programming languages, including Dart used in Flutter.

Importance in Mobile Applications:

Dynamic Data Handling: RESTful APIs allow mobile applications to retrieve and update data in real-time without needing to hardcode it into the app. This makes apps more flexible and user-friendly.

Server Communication: Mobile apps often need to interact with a backend server to perform tasks like user authentication, data storage, or content retrieval. REST APIs make this possible in a standardized way.

Scalability: REST APIs are scalable and can handle requests from multiple clients at once, making them ideal for apps that may grow in user base.

Platform Independence: RESTful APIs can be used by any platform that can send HTTP requests, including Android, iOS, and web applications. This allows for consistency across different client apps.

Modularity and Maintainability: Developers can separate the backend logic from the frontend (app UI) using RESTful APIs, making the codebase cleaner and easier to maintain.

**2. Describe how JSON data is parsed and used in Flutter.**

JSON (JavaScript Object Notation) is a lightweight and human-readable data-interchange format. It is commonly used for exchanging data between a client (like a Flutter app) and a server via RESTful APIs

When a Flutter app sends a request to a RESTful API, it often receives the response data in JSON format. Flutter provides tools to decode and use this data in the app.

Steps to Parse and Use JSON in Flutter:

Make an HTTP Request: Use the http package in Flutter to make a network request to the REST API.

Decode the JSON Response: The response is a JSON string. Use the jsonDecode() function from Dart's dart:convert library to convert it into a Dart object like Map or List.

Use the Data: Once converted, the data can be used in the app (e.g., displaying it in UI widgets or storing it in variables).

Model Class (Optional but Recommended): For better code management, create Dart model classes that represent the structure of the JSON data. You can then use fromJson() and toJson() methods to convert between JSON and Dart objects.

import 'dart:convert';

import 'package:http/http.dart' as http;

void fetchData() async {

final response = await http.get(Uri.parse('https://api.example.com/users'));

if (response.statusCode == 200) {

var jsonData = jsonDecode(response.body); // Convert JSON string to Dart object

print(jsonData['name']); // Accessing data

} else {

print('Failed to load data');

}

}

This shows how Flutter interacts with a REST API and uses the received JSON data in the app.

**3. Explain the purpose of HTTP methods (GET, POST, PUT, DELETE) and when to use each.**

HTTP methods define the type of action the client wants to perform on a server resource. These methods are used in RESTful APIs to perform CRUD (Create, Read, Update, Delete) operations.

1. GET Method:

Purpose: To retrieve data from the server.

Use Case: When the application needs to read or fetch information, such as a list of users, messages, products, etc.

Example: Getting the list of all users from the server.

http.get(Uri.parse('https://api.example.com/users'));

2. POST Method:

Purpose: To send new data to the server to create a new resource.

Use Case: When a user signs up, submits a form, or creates a new post or message.

Example: Sending user registration data to the server.

http.post(

Uri.parse('https://api.example.com/register'),

body: jsonEncode({"name": "Karan", "email": "karan@example.com"}),

headers: {"Content-Type": "application/json"},

);

3. PUT Method:

Purpose: To update existing data on the server.

Use Case: When editing a user's profile, updating settings, or changing existing content.

Example: Updating user profile information.

http.put(

Uri.parse('https://api.example.com/users/1'),

body: jsonEncode({"name": "Karan Updated"}),

headers: {"Content-Type": "application/json"},

);

4. DELETE Method:

Purpose: To remove a resource from the server.

Use Case: When deleting a message, a user account, or any record from the system.

Example: Deleting a user from the database.

http.delete(Uri.parse('https://api.example.com/users/1'));