**Section 1: API Fundamentals — 50 Steps**

**Beginner Level (1–20)**

**Step 1:** *What is an API?*  
An API (Application Programming Interface) is a set of rules that lets software applications communicate.

**Step 2:** *Why do we need APIs?*  
They connect different systems so they can share data and services.

**Step 3:** *Real-life analogy for API*  
Like a waiter who takes your order (request) to the kitchen and brings food (response).

**Step 4:** *Main components of an API*

* Endpoint (URL)
* Request
* Response

**Step 5:** *Types of APIs*

1. **Open** — Public access
2. **Internal** — Inside organization
3. **Partner** — Shared with partners
4. **Composite** — Multiple calls in one

**Step 6:** *What is an API endpoint?*  
A unique address where an API resource is accessed.

**Step 7:** *What is an API request?*  
A message asking the API to perform an action or return data.

**Step 8:** *What is an API response?*  
The reply from the API, containing data or status info.

**Step 9:** *Common data formats in APIs*

* JSON (most popular)
* XML
* YAML

**Step 10:** *What is a query parameter?*  
Extra data in a URL to filter results (e.g., ?status=active).

**Step 11:** *What is a path parameter?*  
A variable in the URL path (e.g., /users/{id}).

**Step 12:** *What is a payload?*  
The request body sent to the server, often in POST or PUT.

**Step 13:** *What are request headers?*  
Metadata about the request (e.g., authentication, content type).

**Step 14:** *What are response headers?*  
Metadata about the server’s response (e.g., date, cache rules).

**Step 15:** *What is an API key?*  
A unique code that authenticates the request.

**Step 16:** *What is an SDK?*  
A toolkit that helps developers work with an API.

**Step 17:** *What is an API call?*  
A single request made to an API endpoint.

**Step 18:** *What is an API integration?*  
Connecting two or more systems via APIs.

**Step 19:** *What is API documentation?*  
Instructions explaining how to use the API.

**Step 20:** *What is a REST API?*  
An API following REST architecture, using HTTP methods.

**Intermediate Level (21–35)**

**Step 21:** *What is an RPC API?*  
Remote Procedure Call API — runs functions remotely.

**Step 22:** *What is a SOAP API?*  
An XML-based API using the SOAP protocol.

**Step 23:** *What is synchronous communication?*  
Client waits for the server to respond before continuing.

**Step 24:** *What is asynchronous communication?*  
Client sends request and moves on without waiting.

**Step 25:** *What is API latency?*  
Time taken to get a response from an API.

**Step 26:** *What is API throughput?*  
Number of requests an API can handle per second.

**Step 27:** *What is API uptime?*  
The percentage of time the API is available.

**Step 28:** *What is API caching?*  
Saving responses to speed up future requests.

**Step 29:** *What is pagination in APIs?*  
Breaking large datasets into smaller pages.

**Step 30:** *What is filtering in APIs?*  
Restricting results to match conditions.

**Step 31:** *What is sorting in APIs?*  
Ordering results by a field.

**Step 32:** *What is content negotiation?*  
Deciding data format (JSON/XML) based on request.

**Step 33:** *What is HATEOAS?*  
REST concept of embedding next possible actions as links in a response.

**Step 34:** *What is rate limiting?*  
Restricting how many requests a user can make in a time frame.

**Step 35:** *What is throttling?*  
Slowing down requests to avoid overload.

**Advanced Level (36–50)**

**Step 36:** *What is API versioning?*  
Running multiple versions (e.g., /v1 and /v2).

**Step 37:** *Why version APIs?*  
To upgrade without breaking old clients.

**Step 38:** *What is API security?*  
Protecting APIs using authentication, authorization, and encryption.

**Step 39:** *What is OAuth?*  
A secure protocol for delegated access.

**Step 40:** *What is JWT?*  
JSON Web Token — secure token for authentication.

**Step 41:** *What is mutual TLS?*  
Two-way certificate verification for secure API access.

**Step 42:** *What is API monitoring?*  
Tracking API performance and errors.

**Step 43:** *What is API logging?*  
Recording request and response details.

**Step 44:** *What is an API gateway?*  
A system managing API traffic, security, and routing.

**Step 45:** *What is service discovery?*  
Finding APIs dynamically within a network.

**Step 46:** *What is API mocking?*  
Simulating API responses for testing.

**Step 47:** *What is API deprecation?*  
Phasing out old API versions.

**Step 48:** *What is an API sandbox?*  
A safe testing environment for APIs.

**Step 49:** *What is API orchestration?*  
Combining multiple API calls into one process.

**Step 50:** *Best practices for API design*

* Clear documentation
* Consistent naming
* Secure authentication
* Handle errors well

**Section 2: REST API — 50 Steps**

**Beginner Level (1–20)**

**Step 1:** *What is a REST API?*  
A REST (Representational State Transfer) API is a web service that uses HTTP methods to access and manipulate resources.

**Step 2:** *What are REST resources?*  
Any object or data entity that can be accessed via a URL, like /users or /products.

**Step 3:** *Key principles of REST*

* Client-Server Separation
* Stateless Communication
* Uniform Interface
* Cacheable Responses
* Layered System

**Step 4:** *What is a REST endpoint?*  
A unique URL that represents a specific resource.

**Step 5:** *What is the GET method?*  
Retrieves data without changing it.

**Step 6:** *What is the POST method?*  
Creates a new resource on the server.

**Step 7:** *What is the PUT method?*  
Updates or replaces a resource completely.

**Step 8:** *What is the PATCH method?*  
Partially updates a resource.

**Step 9:** *What is the DELETE method?*  
Removes a resource from the server.

**Step 10:** *What is HTTP status code 200?*  
OK – the request was successful.

**Step 11:** *What is HTTP status code 201?*  
Created – a new resource was successfully added.

**Step 12:** *What is HTTP status code 400?*  
Bad Request – the request is invalid.

**Step 13:** *What is HTTP status code 401?*  
Unauthorized – authentication is required.

**Step 14:** *What is HTTP status code 404?*  
Not Found – the resource does not exist.

**Step 15:** *What is HTTP status code 500?*  
Internal Server Error – something went wrong on the server.

**Step 16:** *What is a REST request header?*  
Metadata like authentication info, content type, and accepted formats.

**Step 17:** *What is a REST response header?*  
Information about the server’s response, such as content length or cache rules.

**Step 18:** *What is a REST request body?*  
The data sent to the server (usually with POST, PUT, or PATCH).

**Step 19:** *What is content negotiation?*  
Choosing the format of the response (JSON, XML, etc.) based on request headers.

**Step 20:** *Why use JSON in REST?*  
It’s lightweight, easy to read, and works well with JavaScript.

**Intermediate Level (21–35)**

**Step 21:** *What is statelessness in REST?*  
The server does not store client session data between requests.

**Step 22:** *What is caching in REST APIs?*  
Storing responses temporarily to improve performance.

**Step 23:** *What is pagination in REST APIs?*  
Breaking large data sets into smaller pages with parameters like ?page=1&limit=50.

**Step 24:** *What is filtering in REST APIs?*  
Restricting results by adding parameters like ?status=active.

**Step 25:** *What is sorting in REST APIs?*  
Ordering results by a specific field, e.g., ?sort=created\_at.

**Step 26:** *What is HATEOAS?*  
Hypermedia links in responses to guide clients on next actions.

**Step 27:** *What are RESTful best practices?*  
Use nouns, proper HTTP methods, and consistent URL structure.

**Step 28:** *What is an idempotent operation in REST?*  
An operation that has the same effect no matter how many times it’s called (e.g., GET, DELETE).

**Step 29:** *What is a safe method in REST?*  
An HTTP method that does not alter server data (e.g., GET, HEAD).

**Step 30:** *What is OPTIONS in REST?*  
A method used to describe the communication options for a resource.

**Step 31:** *What is HEAD in REST?*  
Similar to GET but returns only headers, not the body.

**Step 32:** *What is TRACE in REST?*  
Returns the received request for debugging purposes.

**Step 33:** *What is PATCH vs PUT?*

* PUT replaces the entire resource.
* PATCH updates only parts of the resource.

**Step 34:** *Why should REST APIs use SSL/TLS?*  
To secure data transfer using HTTPS.

**Step 35:** *What is URI vs URL in REST?*

* URI: Identifier for a resource.
* URL: URI with location details for accessing it.

**Advanced Level (36–50)**

**Step 36:** *What is REST API versioning?*  
Supporting multiple API versions, e.g., /v1/users, /v2/users.

**Step 37:** *Why is API versioning important?*  
It prevents breaking older clients when making changes.

**Step 38:** *What is API rate limiting in REST?*  
Restricting request frequency to avoid overloading the API.

**Step 39:** *What is API throttling in REST?*  
Slowing requests intentionally to manage traffic.

**Step 40:** *What is API authentication in REST?*  
Verifying the client’s identity using API keys, OAuth, or JWT.

**Step 41:** *What is REST API authorization?*  
Determining which resources the client can access.

**Step 42:** *What is token-based authentication in REST?*  
Using tokens instead of usernames/passwords for access.

**Step 43:** *What is OAuth 2.0 in REST?*  
A protocol that allows secure delegated access to APIs.

**Step 44:** *What is JWT in REST APIs?*  
A compact, secure token format for authentication.

**Step 45:** *What is REST API logging?*  
Recording requests, responses, and errors for analysis.

**Step 46:** *What is REST API monitoring?*  
Tracking uptime, response time, and error rates.

**Step 47:** *What is REST API mocking?*  
Simulating endpoints for testing without hitting live servers.

**Step 48:** *What is REST API orchestration?*  
Combining multiple API calls into a single aggregated response.

**Step 49:** *What is REST API gateway?*  
A central system to manage requests, routing, and security.

**Step 50:** *Best practices for REST APIs*

* Use proper HTTP methods.
* Keep endpoints consistent.
* Return meaningful error messages.
* Secure with HTTPS.

**Section 3: SOAP Web Services — 50 Steps**

**Beginner Level (1–20)**

**Step 1:** *What is SOAP?*  
SOAP (Simple Object Access Protocol) is a protocol for exchanging structured data between systems using XML.

**Step 2:** *Why use SOAP?*  
It ensures strict standards, strong security, and reliability in communication.

**Step 3:** *What is the SOAP messaging format?*  
SOAP messages are in XML and include an **Envelope**, **Header**, and **Body**.

**Step 4:** *What is a SOAP Envelope?*  
The root element that defines the start and end of the SOAP message.

**Step 5:** *What is a SOAP Header?*  
An optional section for metadata like authentication, transaction IDs, or routing info.

**Step 6:** *What is a SOAP Body?*  
The main section containing the actual request or response data.

**Step 7:** *What is a SOAP Fault?*  
An error message returned when a request cannot be processed.

**Step 8:** *What is WSDL?*  
Web Services Description Language — an XML file that defines the operations and structure of a SOAP service.

**Step 9:** *Why is WSDL important?*  
It allows clients to understand how to call the service and what data is required.

**Step 10:** *What is UDDI?*  
Universal Description, Discovery, and Integration — a registry for finding web services.

**Step 11:** *What protocols can SOAP use?*  
HTTP, HTTPS, SMTP, JMS, and more.

**Step 12:** *What is RPC-style SOAP?*  
Each request is treated as a remote procedure call with parameters.

**Step 13:** *What is Document-style SOAP?*  
Entire XML documents are sent and received instead of direct function calls.

**Step 14:** *What is the difference between SOAP 1.1 and SOAP 1.2?*  
SOAP 1.2 has improved error handling and more standardized processing rules.

**Step 15:** *What is a SOAP client?*  
An application that sends SOAP requests to a SOAP server.

**Step 16:** *What is a SOAP server?*  
A service that receives and processes SOAP requests.

**Step 17:** *What is a SOAP action?*  
A specific header field that indicates the intent of the SOAP request.

**Step 18:** *What is an XSD in SOAP?*  
XML Schema Definition — defines the structure and data types of XML messages.

**Step 19:** *What is SOAP encoding?*  
A set of rules for serializing data types in SOAP messages.

**Step 20:** *What is a SOAP attachment?*  
Binary data sent with a SOAP message, usually using MIME encoding.

**Intermediate Level (21–35)**

**Step 21:** *What is WS-Security in SOAP?*  
A standard for securing SOAP messages with encryption, signatures, and tokens.

**Step 22:** *What is message-level security?*  
Encrypting the actual SOAP message rather than just the transport layer.

**Step 23:** *What is transport-level security?*  
Using HTTPS or other protocols to secure the connection.

**Step 24:** *What is a SOAP intermediary?*  
A system that processes SOAP headers before forwarding the message.

**Step 25:** *What is SOAP routing?*  
Defining how messages travel through intermediaries before reaching the final recipient.

**Step 26:** *What is WS-Addressing?*  
A standard for including routing information in SOAP headers.

**Step 27:** *What is WS-ReliableMessaging?*  
Ensures guaranteed message delivery even in case of network failures.

**Step 28:** *What is WS-AtomicTransaction?*  
Supports distributed transactions across multiple services.

**Step 29:** *What is WS-Policy?*  
A way for services to declare security and reliability requirements.

**Step 30:** *What is SOAP over JMS?*  
Using Java Messaging Service as the transport protocol for SOAP.

**Step 31:** *What is SOAP over SMTP?*  
Sending SOAP messages via email protocols.

**Step 32:** *What is SOAP over TCP?*  
Using TCP for faster SOAP communication within networks.

**Step 33:** *Why is SOAP more secure than REST?*  
SOAP supports WS-Security, XML encryption, and strict schemas.

**Step 34:** *What is schema validation in SOAP?*  
Checking that XML messages follow the rules defined in the XSD.

**Step 35:** *What is the difference between SOAP and XML-RPC?*  
SOAP is more feature-rich with standards like WS-Security; XML-RPC is simpler.

**Advanced Level (36–50)**

**Step 36:** *What is SOAP message optimization?*  
Techniques like MTOM (Message Transmission Optimization Mechanism) to send binary data efficiently.

**Step 37:** *What is MTOM in SOAP?*  
A method for efficiently sending binary data in SOAP messages.

**Step 38:** *What is SwA in SOAP?*  
SOAP with Attachments — another method for sending files with SOAP.

**Step 39:** *What is SOAP binding?*  
Defines how SOAP messages are transmitted over a specific protocol.

**Step 40:** *What is literal vs encoded SOAP binding?*

* **Literal:** XML matches the schema exactly.
* **Encoded:** Uses SOAP encoding rules.

**Step 41:** *What is document-literal-wrapped style?*  
A WSDL style where each operation has a single XML wrapper element.

**Step 42:** *What is WS-Federation?*  
A standard for identity sharing between security domains in SOAP services.

**Step 43:** *What is WS-Trust?*  
Defines extensions for issuing, renewing, and validating security tokens.

**Step 44:** *What is WS-SecureConversation?*  
Establishes a secure communication session between client and server.

**Step 45:** *What is SOAP fault code vs fault string?*

* **Fault Code:** Type of error.
* **Fault String:** Human-readable error message.

**Step 46:** *What is SOAP logging?*  
Recording SOAP request and response messages for debugging.

**Step 47:** *What is SOAP message replay protection?*  
Preventing malicious re-use of captured SOAP messages.

**Step 48:** *What is a SOAP load test?*  
Simulating high traffic to measure SOAP service performance.

**Step 49:** *What is SOAP service mocking?*  
Creating fake SOAP endpoints for development/testing.

**Step 50:** *Best practices for SOAP services*

* Always validate XML against XSD.
* Use WS-Security for sensitive data.
* Keep WSDLs updated.
* Log errors with enough context.

**Section 4: API Security & Management — 50 Steps**

**Beginner Level (1–20)**

**Step 1:** *What is API security?*  
The practice of protecting APIs from unauthorized access, misuse, and attacks.

**Step 2:** *Why is API security important?*  
APIs often handle sensitive data, so they’re prime targets for hackers.

**Step 3:** *What is authentication in APIs?*  
Verifying the identity of the user or system making the request.

**Step 4:** *What is authorization in APIs?*  
Determining what resources an authenticated user can access.

**Step 5:** *What is an API key?*  
A unique identifier used for authentication and tracking API usage.

**Step 6:** *Why should API keys be kept secret?*  
If leaked, attackers can make requests as if they were you.

**Step 7:** *What is Basic Authentication?*  
Using a username and password encoded in Base64 for access.

**Step 8:** *What is OAuth 2.0?*  
A protocol that allows secure delegated API access without sharing credentials.

**Step 9:** *What is JWT (JSON Web Token)?*  
A compact, signed token used for secure API authentication.

**Step 10:** *What is mutual TLS?*  
Both client and server verify each other’s identity using SSL certificates.

**Step 11:** *What is HTTPS?*  
A secure version of HTTP that encrypts data in transit.

**Step 12:** *Why use HTTPS for APIs?*  
It protects against eavesdropping and man-in-the-middle attacks.

**Step 13:** *What is API rate limiting?*  
Restricting the number of requests per user or app in a given timeframe.

**Step 14:** *What is API throttling?*  
Slowing down request processing to prevent overload.

**Step 15:** *What is IP whitelisting?*  
Allowing API access only from approved IP addresses.

**Step 16:** *What is a CORS policy?*  
A security rule controlling which domains can call your API.

**Step 17:** *What is API versioning for security?*  
Maintaining multiple versions so older ones can be retired safely.

**Step 18:** *What is API logging?*  
Recording API requests, responses, and errors for security analysis.

**Step 19:** *What is API monitoring?*  
Tracking API performance, uptime, and potential attacks.

**Step 20:** *What is an API gateway?*  
A management tool that controls, secures, and routes API traffic.

**Intermediate Level (21–35)**

**Step 21:** *What is an API token?*  
A secure, temporary key for authentication, often short-lived.

**Step 22:** *What is token expiration?*  
Setting tokens to expire after a certain time for better security.

**Step 23:** *What is token refresh?*  
Requesting a new token without logging in again.

**Step 24:** *What is HMAC authentication?*  
Using a hashed message authentication code for request verification.

**Step 25:** *What is API encryption at rest?*  
Storing sensitive data in encrypted form.

**Step 26:** *What is API encryption in transit?*  
Encrypting data while it’s moving between systems.

**Step 27:** *What is payload signing?*  
Digitally signing API request data to prevent tampering.

**Step 28:** *What is API replay protection?*  
Blocking repeated use of captured API requests.

**Step 29:** *What is a security token service (STS)?*  
A system that issues security tokens for API access.

**Step 30:** *What is role-based access control (RBAC)?*  
Restricting API access based on user roles.

**Step 31:** *What is attribute-based access control (ABAC)?*  
Restricting access based on user attributes (e.g., department).

**Step 32:** *What is API anomaly detection?*  
Identifying unusual traffic patterns that may indicate an attack.

**Step 33:** *What is API abuse detection?*  
Finding and blocking malicious API usage.

**Step 34:** *What is an API firewall?*  
A security layer filtering and blocking suspicious requests.

**Step 35:** *What is API DDoS protection?*  
Defending against Distributed Denial of Service attacks targeting APIs.

**Advanced Level (36–50)**

**Step 36:** *What is zero trust security for APIs?*  
Never trust, always verify — even for internal API calls.

**Step 37:** *What is API penetration testing?*  
Simulating attacks to find vulnerabilities in APIs.

**Step 38:** *What is API fuzz testing?*  
Sending random or invalid inputs to see how the API reacts.

**Step 39:** *What is API schema validation?*  
Ensuring requests follow the defined schema to avoid injection attacks.

**Step 40:** *What is API threat modeling?*  
Analyzing possible attack paths and weaknesses in an API.

**Step 41:** *What is API security automation?*  
Using tools to automatically check API vulnerabilities.

**Step 42:** *What is API governance?*  
Defining security, usage, and quality rules for APIs across an organization.

**Step 43:** *What is API SLA (Service Level Agreement)?*  
A contract defining API availability, performance, and support guarantees.

**Step 44:** *What is API observability?*  
Gaining insight into API health, performance, and failures.

**Step 45:** *What is API key rotation?*  
Regularly changing keys to reduce risk from leaks.

**Step 46:** *What is API secrets management?*  
Storing keys, passwords, and tokens securely using vaults.

**Step 47:** *What is API incident response?*  
The process of investigating and mitigating API security breaches.

**Step 48:** *What is API compliance?*  
Meeting legal and regulatory standards like GDPR or HIPAA.

**Step 49:** *What is API vulnerability scanning?*  
Automatically finding security flaws in APIs.

**Step 50:** *Best practices for API security*

* Use HTTPS everywhere.
* Apply least privilege access.
* Monitor and log all requests.
* Rotate keys and tokens regularly.
* Validate all inputs.

**Section 5: Advanced API Concepts — 50 Steps**

**Beginner-to-Intermediate (1–20)**

**Step 1:** *What is an API design pattern?*  
A reusable solution to common API development problems.

**Step 2:** *What is RESTful design?*  
Using HTTP methods, clear endpoints, and stateless communication.

**Step 3:** *What is API-first development?*  
Designing the API before building the actual application.

**Step 4:** *What is API-as-a-Product?*  
Treating the API like a product, with documentation, support, and monetization.

**Step 5:** *What is API monetization?*  
Charging for API usage via subscription, pay-per-use, or freemium models.

**Step 6:** *What is API discoverability?*  
Making APIs easy for developers to find and understand.

**Step 7:** *What is API scalability?*  
Designing APIs to handle increasing load without performance loss.

**Step 8:** *What is API maintainability?*  
Making APIs easy to update, fix, and improve.

**Step 9:** *What is API reusability?*  
Designing APIs so they can be reused across multiple applications.

**Step 10:** *What is GraphQL?*  
An API query language that lets clients request exactly the data they need.

**Step 11:** *What is GraphQL mutation?*  
A way to change data in a GraphQL API.

**Step 12:** *What is GraphQL subscription?*  
A real-time data update mechanism in GraphQL.

**Step 13:** *What is gRPC?*  
A high-performance API framework using Protocol Buffers for serialization.

**Step 14:** *What is Protocol Buffers (Protobuf)?*  
A binary serialization format used by gRPC for faster communication.

**Step 15:** *What is an API facade?*  
A single API that hides complex backend systems from the client.

**Step 16:** *What is an API adapter?*  
A layer that translates requests between incompatible systems.

**Step 17:** *What is an API proxy?*  
A middle layer that forwards API requests and responses.

**Step 18:** *What is a composite API?*  
An API that combines responses from multiple services into one.

**Step 19:** *What is API chaining?*  
Executing multiple API calls in sequence to complete a process.

**Step 20:** *What is API orchestration?*  
Coordinating multiple APIs to work together as one workflow.

**Intermediate-to-Advanced (21–35)**

**Step 21:** *What is API virtualization?*  
Creating simulated APIs for testing when the real ones are unavailable.

**Step 22:** *What is API load balancing?*  
Distributing requests across multiple servers for performance.

**Step 23:** *What is API failover?*  
Switching to backup servers if the main one fails.

**Step 24:** *What is API blue-green deployment?*  
Switching traffic between two identical environments for updates with zero downtime.

**Step 25:** *What is API canary release?*  
Rolling out new API features to a small group before full deployment.

**Step 26:** *What is API schema evolution?*  
Updating API data models without breaking clients.

**Step 27:** *What is backward compatibility in APIs?*  
Ensuring new changes don’t break existing client applications.

**Step 28:** *What is forward compatibility in APIs?*  
Ensuring old clients can still work with future changes.

**Step 29:** *What is API sandboxing?*  
Providing a safe test environment for developers.

**Step 30:** *What is an API lifecycle?*  
Stages: Design → Develop → Test → Deploy → Monitor → Retire.

**Step 31:** *What is event-driven API architecture?*  
APIs that trigger actions when specific events occur.

**Step 32:** *What is webhook vs polling?*

* **Webhook:** Server pushes updates to client.
* **Polling:** Client repeatedly checks for updates.

**Step 33:** *What is API schema registry?*  
A centralized place to store and manage API schemas.

**Step 34:** *What is hypermedia API?*  
APIs that include navigational links in responses (HATEOAS).

**Step 35:** *What is API telemetry?*  
Collecting real-time metrics from API usage.

**Advanced Concepts (36–50)**

**Step 36:** *What is API microgateway?*  
A lightweight gateway deployed close to services for local API control.

**Step 37:** *What is API mesh?*  
A network of interconnected APIs managed as a whole.

**Step 38:** *What is API federation?*  
Merging multiple APIs into a single unified API.

**Step 39:** *What is API discoverability in microservices?*  
Using service discovery to locate APIs in a distributed environment.

**Step 40:** *What is API sharding?*  
Splitting API data or traffic across multiple servers for efficiency.

**Step 41:** *What is multi-tenancy in APIs?*  
Supporting multiple separate customers within a single API instance.

**Step 42:** *What is API analytics?*  
Tracking API usage trends for optimization and business insights.

**Step 43:** *What is AI-powered API optimization?*  
Using machine learning to improve API performance and scaling.

**Step 44:** *What is API chaos testing?*  
Intentionally causing failures to test API resilience.

**Step 45:** *What is API self-healing?*  
Automatically recovering from failures using automation.

**Step 46:** *What is API contract testing?*  
Verifying that API responses meet agreed-upon specifications.

**Step 47:** *What is API schema linting?*  
Checking API specifications for consistency and errors.

**Step 48:** *What is API monetization platform?*  
A tool for managing API subscriptions, billing, and payments.

**Step 49:** *What is API sustainability?*  
Designing APIs to minimize energy consumption and infrastructure costs.

**Step 50:** *Best practices for advanced APIs*

* Design for scale from the start.
* Monitor continuously.
* Use automated tests.
* Keep security in mind at all times.