

Basics / HTTP Concepts

1. Common HTTP Methods in API?

The most common HTTP methods are GET (retrieve data), POST (create new resource), PUT (full update), PATCH (partial update), and DELETE (remove resources).

2. Different common status codes in API? (2xx, 3xx, 4xx, 5xx)

2xx Success

200 OK – Request succeeded and returned data.

201 Created – New resource was created successfully.

204 No Content – Request succeeded but no data returned.

3xx Redirection

301 Moved Permanently – Resource permanently moved to a new URL.

302 Found – Resource temporarily available at another URL.

304 Not Modified – Resource hasn't changed; cached version can be used.

4xx Client Error

400 Bad Request – Request is invalid or malformed.

401 Unauthorized – Authentication is missing or failed.

403 Forbidden – Access is denied despite authentication.

404 Not Found – Requested resource does not exist.

5xx Server Error

500 Internal Server Error – Server encountered an unexpected error.

502 Bad Gateway – Server got an invalid response from another server.

503 Service Unavailable – Server is overloaded or under maintenance.

3. Difference between PUT and PATCH?

PUT updates the entire resource, while **PATCH** updates only specific fields of a resource.

4. Query Params vs Path Params (Interview-style short answer):

Path parameters identify a specific resource, while **query** parameters filter, sort, or modify the response.

5. And can you use PUT before POST? Which comes first?

Normally **POST** comes first to create a resource, and **PUT** is used afterward to update it.

6. **What does 409 mean?**

409 Conflict means the request failed due to a conflict with the current state of the resource.

Authorization & Security

1. **Authorization in automation testing?**

Authorization in automation testing is **verifying that a user has the correct permissions to access or perform actions on an API/application.**

2. **Imagine you are testing a login page, how would you ensure that it is secure against all invalid data (without using automation)?**

I perform **manual negative testing** using invalid credentials, empty fields, special characters, SQL injection, and boundary values, ensuring **secure error messages without exposing system details.**

3. **"Equivalence Partitioning" – How does this test design technique help in testing the login page?**

Equivalence Partitioning **reduces test effort by grouping inputs into valid and invalid classes and testing one value from each group.**

4. **About Maximum and minimum input values, to which field were you referring? Username or password?**

Maximum and minimum value validation applies to **both username and password fields** to enforce length and format constraints.

Negative Scenarios & Error Handling

5. **How do you handle negative scenarios in API?**

I send **invalid, missing, or unexpected inputs and expired tokens**, then verify correct **error status codes (400, 401, 403, 404, 409)** and secure error messages.

6. **And you want to put those negative scenario data in a database using API. So what steps are you going to do?**

I design negative test cases, send invalid inputs via API, capture responses, and **verify that error details are correctly logged/stored in the database.**

7. **If you are running the POST command and it is failing, what might be the issues?**

POST failures may occur due to **invalid payloads, missing mandatory fields, wrong content-type, authentication issues, or server validation errors.**

8. **In a front-end, you see HTTP 200 OK, but it retrieves zero records from your DB. How will you resolve this issue?**

I verify **request parameters, API filtering logic, and database queries** to identify data mismatches.

Dynamic Values & Schema Validation

1. **If you are working with dynamic values in API, what steps will you follow?**
I capture **dynamic values from a previous response, store them as variables, and reuse them in subsequent requests.**
 2. **About schema validations? What about nested JSON objects?**
Schema validation ensures **correct JSON structure, data types, and mandatory fields**, including validation of **nested JSON hierarchies.**
 3. **If you are performing API testing, what information do you get from the developer?**
I receive **API documentation/specs including endpoints, methods, payloads, headers, authentication, and expected responses.**
-

Testing Approach & Scenarios

1. **If you have created 10–20 test cases, how many would you automate vs manual?**
I automate **70–80% stable, repeatable tests** and keep **20–30% exploratory or unstable tests manual.**
2. **What kind of tests do you perform in REST API? Provide examples and explain your approach.**
I perform **functional, negative, validation, security, performance, and integration testing** to verify correctness, reliability, and scalability.
3. **How do you design a test case or a test suite for API or any web applications?**
I analyze requirements, define scenarios, prepare test data, and design **positive, negative, and edge case coverage.**
4. **How did you perform integration testing in your API automation framework?**
I chain related APIs, pass dynamic data between them, and validate **end-to-end workflows.**

5. **Have you extended support for production validation? Give examples.**
Yes, I perform **post-deployment smoke and sanity tests** on critical APIs in production.

Challenges & Reporting

1. **What are the challenges you faced in API testing?**
Challenges include **unclear documentation, authentication complexity, unstable environments, dynamic data handling, and service dependencies.**
2. **If there is a conflict with a developer about a defect, how would you approach it?**
I present reproducible evidence, align with requirements, and discuss collaboratively before escalation.
3. **Suppose you need to report a bug at the last moment, what should your defect report include?**
A defect report must include **steps to reproduce, environment, expected vs actual results, logs/screenshots, and severity.**

Broken Links

1. **How do you find broken links (Dead Links) in API?**
I validate all endpoints by checking **HTTP status codes** and flag **4xx/5xx errors like 404 or 410** as dead links.
2. **How do you find broken links (Dead Links) in UI?**
I extract UI hyperlinks and verify their **HTTP responses manually or via automation tools.**

✓ API Testing – Theory Based Questions (1–25)

1. **What is API testing?**
API testing validates functionality, reliability, performance, and security by sending requests and verifying responses without using UI.
2. **Why is API testing important?**
API testing detects issues early, ensures backend stability, and improves overall system reliability.
3. **Difference between API testing and UI testing?**
API testing is faster and validates business logic without UI dependency, while UI testing validates user workflows but is slower and UI-dependent.

4. **What are HTTP methods used in API testing?**

The main HTTP methods are GET, POST, PUT, PATCH, and DELETE for CRUD operations.

5. **What is REST API?**

REST API is a stateless architectural style that uses HTTP methods to perform CRUD operations.

6. **What is SOAP API?**

SOAP is an XML-based protocol with strict standards and built-in security for structured messaging.

7. **Difference between REST and SOAP?**

REST is lightweight, fast, and supports JSON/XML, while SOAP is heavier, XML-only, and can be stateful.

8. **What is JSON?**

JSON is a lightweight data format used to exchange structured data between client and server.

9. **What are status codes?**

HTTP status codes indicate request results: 2xx success, 3xx redirection, 4xx client errors, and 5xx server errors.

10. **What is authentication in API?**

Authentication verifies user identity using OAuth, JWT, API keys, or Basic Auth.

11. **What is authorization?**

Authorization determines user permissions after successful authentication.

12. **What is API endpoint?**

An API endpoint is a specific URL where requests are sent to access resources.

13. **What is payload?**

Payload is the data sent in the body of an API request.

14. **What is idempotency?**

Idempotency means repeated API calls produce the same result, such as with PUT requests.

15. **What is rate limiting?**

Rate limiting controls how many API requests can be made within a time period.

16. **What is API versioning?**

API versioning manages changes by maintaining versions like v1 and v2.

17. What tools have you used for API testing?

Common tools include Postman, Pytest, REST Assured, k6, Swagger, and Jenkins.

18. What is contract testing?

Contract testing ensures APIs follow agreed schemas between client and server.

19. What is mocking in API testing?

Mocking simulates API responses when backend services are unavailable.

20. What is schema validation?

Schema validation verifies that responses match the expected JSON structure and data types.

21. What is API automation?

API automation uses scripts to execute repeated API tests automatically.

22. What is CI/CD in API testing?

CI/CD integrates automated API tests into build pipelines.

23. What is API security testing?

API security testing validates authentication, authorization, and encryption mechanisms.

24. What is performance testing in API?

Performance testing measures API load, stress handling, and scalability.

25. What is negative testing?

Negative testing sends invalid inputs to verify proper error handling.

✓ Scenario-Based API Testing Questions (26–50)

26. API returns 200 but wrong data. What do you do?

I validate business logic, request payload, database records, and logs.

27. API is slow. How do you debug?

I analyze response time, server performance, and network latency.

28. Login API failing intermittently. Approach?

I check token expiry, session handling, and server logs.

29. You get 401 error. What does it mean?

401 indicates authentication failure or invalid credentials.

30. How do you test pagination?

I verify page size, navigation behavior, and boundary cases.

31. API returns 500 error. Next steps?

I review server logs, payload validation, and backend stability.

32. How do you test file upload API?

I validate file type, size limits, and upload integrity.

33. How do you handle dynamic tokens?

I store and reuse tokens using environment variables or scripts.

34. How do you validate response schema?

I use JSON schema validation tools to verify structure.

35. API works in Postman but fails in automation. Why?

This usually occurs due to environment configuration or header mismatches.

36. How do you test API security?

I test authentication bypass, injection risks, and encryption.

37. API works locally but fails in Jenkins. Why?

It is often caused by environment configuration or dependency issues.

38. How do you test API error handling?

I send invalid inputs and verify correct error responses.

39. How do you ensure data consistency?

I cross-verify API responses with database records.

40. How do you test concurrent users?

I simulate load using tools like k6.

41. API response format changed unexpectedly. Action?

I raise a defect and update the API contract.

42. How do you test backward compatibility?

I run regression tests on older API versions.

43. API timeout occurs. Approach?

I check server capacity and retry mechanisms.

44. How do you test caching?

I verify repeated requests return cached responses correctly.

45. API has dependency on another service. How test?

I use mocks or stubs to isolate dependencies.

46. How do you validate headers?

I verify content-type, authorization, and caching headers.

47. How do you prioritize API tests?

I prioritize based on critical business functionality.

48. How do you test bulk data API?

I validate performance, limits, and memory handling.

49. API returns inconsistent results. What next?

I investigate race conditions and backend processing issues.

50. How do you design a scalable API automation framework?

I design modular, reusable frameworks integrated with CI/CD.
