API Testing

Short notes for beginners · Postman · REST · QA · backend freshers

Uncodemy Handwritten Notes · free for students

1. API Basics

- **API** (Application Programming Interface) is a contract that lets one software talk to another. A client sends a **request**, the server sends back a **response**. Example: a weather app asks a weather API for today's data.
- **API testing** checks the business logic layer directly, without a UI. It is faster, more stable and finds bugs earlier than UI tests. In the test pyramid, many unit tests, a good number of API tests, few UI tests.

Type	Key points
REST	Uses HTTP, resources and URLs, usually JSON. Most popular
SOAP	XML only, strict WSDL contract, built in security (WS-Security)
GraphQL	One endpoint, client asks exactly the fields it needs
gRPC	Binary protocol buffers over HTTP/2, very fast, microservices
WebSocket	Two way persistent connection (chat, live data)

- **Endpoint** = full URL of a resource, e.g. https://api.shop.com/v1/users/5. **Base URL** = common start of all endpoints. Data formats: **JSON** (key: value pairs, lightweight) and **XML** (tags).

EXAMPLE JSON body: { "name": "Asha", "age": 24, "skills": ["QA", "API"], "active": true }. Types: string, number, boolean, null, object { }, array [ ].

2. REST and HTTP Methods

- **REST** (Representational State Transfer) principles: client-server, stateless, cacheable, uniform interface, layered system (and optional code on demand). **Stateless** means each request carries everything the server needs.

- Resource naming: use nouns, plural, lowercase: /users, /users/5/orders. Avoid verbs in URLs (not /getUsers). Use versioning like /v1/.

Method	Purpose (CRUD)	Safe	Idempotent
GET	Read data	Yes	Yes
POST	Create a new resource	No	No
PUT	Replace a full resource	No	Yes
PATCH	Update part of a resource	No	Not guaranteed
DELETE	Remove a resource	No	Yes
HEAD / OPTIONS	Headers only / allowed methods	Yes	Yes

- **Safe** = does not change data. **Idempotent** = calling it many times gives the same final state. Calling DELETE /users/5 twice leaves the same state (user gone); calling POST /users twice creates two users.
- Common features: **query params** (?page=2&limit=10), **path params** (/users/5), pagination, filtering, sorting, rate limiting, CORS, HATEOAS links.

EXAMPLE Pagination: page = 3, limit = 20 → the API skips (3 - 1) x 20 = 40 records and returns records 41 to 60.

3. Request and Response Anatomy

- A **request** has: method + URL, headers, optional query / path params, optional body. A **response** has: status line (code + message), headers, body.

Header	Meaning
Content-Type	Format of the body sent: application/json, multipart/form-data
Accept	Format the client wants back
Authorization	Credentials: Basic, Bearer token, API key
User-Agent	Client software name
Cache-Control	Caching rules
Cookie / Set-Cookie	Session data sent / set by the server

- Body types: raw JSON, form-data (files), x-www-form-urlencoded (HTML form style), binary. For POST / PUT with JSON, set the header Content-Type:

application/json.

- Response time is part of quality: most good APIs reply under 500 ms (typical target, depends on the project). Also check response size, headers and body fields.

4. HTTP Status Codes

Code	Name	When used
200	OK	Request succeeded (GET, PUT)
201	Created	New resource created (POST)
202	Accepted	Accepted, processing later
204	No Content	Success with no body (DELETE)
301 / 302	Redirect	Moved permanently / temporarily
304	Not Modified	Use the cached copy
400	Bad Request	Invalid syntax or data
401	Unauthorized	Missing or invalid login / token
403	Forbidden	Logged in but no permission
404	Not Found	Resource does not exist
405	Method Not Allowed	Wrong method for this endpoint
409	Conflict	Duplicate or state conflict
415 / 422	Unsupported / Unprocessable	Wrong media type / validation error
429	Too Many Requests	Rate limit crossed
500	Internal Server Error	Server bug
502 / 503 / 504	Bad Gateway / Unavailable / Timeout	Upstream or server down or slow

- Class: 1xx information, 2xx success, 3xx redirection, 4xx client error, 5xx server error. 401 = who are you? 403 = I know you, but you cannot do this.

5. Authentication and Security

Method	How it works
Basic Auth	username:password in base64 in the header. Not encrypted, use only over HTTPS
API key	Secret key in header or query (x-api-key)
Bearer token / JWT	Authorization: Bearer <token>, token proves identity
OAuth 2.0	Delegated access with access and refresh tokens
Session cookie	Server stores session, client sends the cookie

- **JWT** (JSON Web Token) = header.payload.signature, each base64url encoded. It is signed, not secret, so do not put passwords inside.

EXAMPLE Basic auth header: user:pass → base64 → Authorization: Basic dXNlcjpwYXNz. Base64 can be decoded easily, so it is encoding, not encryption.

- Security tests: missing or expired token (401), wrong role (403), SQL / script injection in fields, sensitive data in URLs or responses, rate limiting, HTTPS only, no stack traces in errors. Top risks list: OWASP API Security Top 10.

6. Postman Basics

- **Postman** is a tool to build, send and test API requests. Structure: Workspace → **Collection** (group of requests, with folders) → Request. Save, share, document and run them.
- **Variables** have scopes: global, collection, environment, data and local. Use them with double curly braces. Switch Dev / QA / Prod using **Environments**.

EXAMPLE Set variable baseUrl = https://reqres.in/api in an environment. Request URL: {{baseUrl}}/users?page=2. Change environment and every request points to the new server.

- Steps to send a request: choose method → enter URL → add Params / Headers / Auth → add Body → Send → check Status, Time, Size, Body (Pretty, Raw, Preview).

- Other features: pre-request script (runs before), Tests tab (runs after), Collection Runner (run all with data files CSV / JSON), mock servers, monitors, API documentation, Newman (command line runner), Postman Console for debugging.
- Practice APIs (free): reqres.in, jsonplaceholder.typicode.com, restful-booker.herokuapp.com, petstore.swagger.io.

7. Writing Tests in Postman

- Tests are written in JavaScript with the pm object and Chai style assertions, in the Tests tab.

EXAMPLE pm.test("Status is 200", function () { pm.response.to.have.status(200); }); var d = pm.response.json(); pm.expect(d.data.id).to.eql(2);

Check	Snippet idea
Status code	pm.response.to.have.status(201)
Response time	pm.expect(pm.response.responseTime).to.be.below(500)
Header present	pm.response.to.have.header("Content-Type")
Body value	pm.expect(jsonData.name).to.eql("Asha")
Save for next request	pm.environment.set("token", jsonData.token)

- **Chaining requests**: login request saves the token in a variable; later requests use {{token}} in the Authorization header. **Data driven runs**: run one request with many rows from a CSV file in the Collection Runner.
- Newman: newman run collection.json -e env.json runs tests from the command line, so they can run in Jenkins or GitHub Actions on every build.

8. What to Test and Tools

EXAMPLE POST /users checks: valid data → 201 + id. Name missing → 400. Duplicate email → 409. No token → 401. Wrong role → 403. Invalid JSON → 400. Very long name → 400 or 422. Wrong method → 405.

- Test types: **functional** (correct output), **validation** (schema, data types, mandatory fields), **negative and boundary**, **security**, **performance / load**, **integration**, **contract** (consumer and provider agree), **regression**.
- Check always: status code, response body fields and types, headers, response time, database change after POST / PUT / DELETE, error message format, and idempotency.

Tool	Use
Postman / Insomnia	Manual and automated API tests with a UI
REST Assured	Java library for API automation
Karate	Simple DSL for API tests and BDD
Python requests + pytest	Scripted API tests in Python
JMeter / k6	Load and performance testing
Swagger / OpenAPI	API documentation and contract
SoapUI	SOAP and REST testing

- REST Assured style: given().baseUri(url).when().get("/users/2").then().statusCode(200).body("data.id", equalTo(2)); Given = setup, When = action, Then = verify.

9. REST vs SOAP and Good Practice

Point	REST	SOAP
Format	JSON, XML, others	XML only
Protocol	HTTP style (architecture)	Strict protocol, own rules
Contract	OpenAPI (optional)	WSDL (must)
Speed	Lighter and faster	Heavier
Use	Web and mobile apps	Banking, enterprise with strict rules

- Test with clean, repeatable data, use separate test environments, never use production data, keep secrets in environment variables, and give every test a clear name.
- Test checklist for a new API: read the documentation (Swagger) → list endpoints and methods → write positive and negative cases → automate in Postman → add to CI → monitor.
- Bug report for an API: endpoint, method, request headers and body, actual response, expected response, steps and environment. Attach cURL or Postman collection.

10. Most Asked Questions (practice)

1. What is an API? Why do we do API testing?
2. Explain REST principles. REST vs SOAP.
3. Difference between PUT and PATCH. What is idempotency?
4. Explain status codes 200, 201, 204, 400, 401, 403, 404, 500.
5. Difference between 401 and 403. Authentication vs authorisation.
6. What are headers, query params and path params?
7. How do you write tests and use variables in Postman?
8. How do you chain two requests (login → get profile)?
9. How do you run Postman tests in CI? What is Newman?
10. Write test cases for a POST /users API.

TIP Interviews: practise on reqres.in, explain one API you tested end to end, and learn JSON, status codes and basic SQL, because they are asked in almost every QA round.