# **Creating API testing resources**

# The user wants a list of APIs for a QA to test and an assignment. I'll suggest suitable free public APIs, such as JSONPlaceholder, ReqRes, OpenWeatherMap, PokeAPI, Swagger Petstore, The Star Wars API (SWAPI), and NASA API for beginners. For the assignment, the QA could create test cases for GET/POST/PUT/DELETE, error handling, authentication, rate limiting, and performance. This ties into the user's focus on testing fundamentals like analytical skills, bug-cracking, and good test planning.

# Here’s a two-part plan to get your QA up to speed on API testing:

# 1. Suggested Public APIs to Explore

# JSONPlaceholder

# – Fake REST API with /posts, /comments, /albums, /todos, /users endpoints

# – Perfect for practicing CRUD & schema validation

# – Docs: <https://jsonplaceholder.typicode.com/>

# ReqRes ([reqres.in](http://reqres.in/))

# – Demo user registration, login, list & single-user endpoints

# – Good for practicing authentication flows & error code checks

# – Docs: <https://reqres.in/>

# Swagger Petstore

# – Classic “pet store” API with full CRUD on /pet, /store, /user

# – Includes file upload, parameterized queries & OAuth2 sandbox

# – Docs: <https://petstore.swagger.io/>

# The Star Wars API (SWAPI)

# – Explore GET calls on /people, /planets, /films, etc.

# – Great for chaining requests and pagination testing

# – Docs: <https://swapi.dev/>

# OpenWeatherMap (free tier)

# – Requires API key; GET current weather, forecasts, geocoding

# – Practice key-based auth, rate-limit & performance checks

# – Docs: <https://openweathermap.org/api>

# GitHub REST API (public endpoints)

# – List repos, users, issues; practice OAuth tokens & header validation

# – Docs: <https://docs.github.com/en/rest>

# 2. API Testing Assignment

# Background:

# You’ll build a mini test suite against the Swagger Petstore API to validate its CRUD, error-handling, performance, and security behaviors.

# a) Test Plan & Scope

# Objectives:

# Verify the correctness of POST/GET/PUT/DELETE /pet workflows

# Validate response schemas, HTTP codes & headers

# Perform negative tests (invalid IDs, missing payload fields)

# Measure response times and observe rate-limit behavior

# Check basic security (SQL injection payloads, malformed JSON)

# In Scope:

# /pet endpoints only

# Functional, negative, performance and security smoke tests

# b) Test Cases (examples)

# Create Pet

# Valid payload → expect 200 OK, body matches request, id generated

# Invalid payload (missing name) → expect 400 Bad Request

# Retrieve Pet

# Valid id → 200 OK, JSON schema match

# Non-existent id → 404 Not Found

# Update Pet

# Change status from "available" to "sold" → 200 OK, verify update

# Send malformed JSON → 415 Unsupported Media Type or 400

# Delete Pet

# Valid id → 200 OK, subsequent GET → 404

# Delete already-deleted id → 404 Not Found

# Performance

# Send 20 concurrent GET /pet/{id} → measure 95th-percentile latency <200 ms

# Security Smoke

# Inject '; DROP TABLE pets;-- in name → expect proper sanitization, no server error

# c) Tools & Setup

# Postman (or REST Client plug-in) for manual execution

# Newman for running collections in CI

# JSON schema validator (built into Postman tests)

# JMeter/Artillery for simple performance checks

# d) Deliverables

# Test Plan document with scope, objectives, environment, data

# Postman collection containing all test cases with inline assertions

# Test report summarizing pass/fail, response times, defects found

# Bug reports for any failures, using a template that includes:

# Steps to reproduce

# Request/response payloads

# Expected vs. actual

# Severity & priority

# This exercise will teach her to design robust test cases, automate them, perform performance checks, and report defects clearly, building strong API testing fundamentals.

# **API Testing Assignment - Deliverables**

**1. Test Plan**

**Testing Objective:** The /pet part of the Swagger Petstore API. We can add, update, get, and delete pets properly. Also, to ensure it handles errors, performs well, and doesn't break if something weird is sent to it.

**What’s included:**

* Test creating, getting, updating, and deleting a pet (CRUD)
* Try with wrong data to see how it reacts
* Check how fast it responds
* Try some basic security stuff (SQL injection)

**Where we’re testing:**

* URL: https://petstore.swagger.io/v2
* Tool: Postman

**Test data examples:**

* Valid pet ID: 123456
* Invalid pet ID: 999999999
* SQL Injection: "DROP TABLE pets"
* Missing name test: { "status": "available" }

**----------------------------------------------------------------------------------------------------------**

**2. Postman Collection**

* <https://zoya-3668638.postman.co/workspace/zoya's-Workspace~75af44f9-5353-4682-8aba-0b528ec10e17/collection/44329802-f9823beb-26bf-4c05-8275-946863887121?action=share&creator=44329802>
* Has built-in test scripts using pm.test()
* Includes tests for: create, get, update, delete, negative, performance, and security

**------------------------------------------------------------------------------------------------------------**

**3. Quick Test Summary**

| **What We Tested** | **Status** | **Response time**  **(ms)** | **Result** |
| --- | --- | --- | --- |
| Create Pet | 200 - Passed | 323 | Pet created and ID saved |
| Get Pet by ID | 200 - Passed | 301 | Correct pet info returned |
| Update Pet | 200 - Passed | 303 | Status changed to "sold" |
| Delete Pet | 200 - Passed | 323 | Pet removed successfully |
| Negative test - Invalid ID | 404 - Passed | 427 | Got proper error message |
| Negative test - Missing fields | 200 - Passed | 424 | Status changed to "available." |
| Performance test - Response time | 200 - Failed | 289 | All above 200ms |
| SQL Injection | 200 - Passed | 277 | No crash or weird behavior |