HTTP IS NOT JUST STATUS CODES

HTTP is a **contract between client and server**, not just a response code and JSON body.
Real API failures happen because testers ignore headers, retries, async behavior, and state guarantees.

---

1. WHAT HTTP REALLY CONSISTS OF

An HTTP request has **four parts**:

1. **Method** – Action you want to perform
   (GET, POST, PUT, PATCH, DELETE)
2. **URL** – Target resource
3. **Headers** – Rules, identity, behavior control
   (Most critical part in real systems)
4. **Body** – Data payload (optional)

Beginner mistake:
Only validating **response body + status code**.

---

2 HEADERS THAT ACTUALLY MATTER IN PRODUCTION

Authorization Header
Authorization: Bearer <token>

Purpose

- Identifies the client/user
- Defines permissions (scopes/roles)

What to test

- Missing token → 401 Unauthorized
- Invalid token → 401
- Expired token → 401
- Insufficient permission → 403 Forbidden

A 200 OK does **not** mean authorization is correct.
Permissions must be validated separately.

Idempotency-Key (CRITICAL FOR PAYMENTS)
Idempotency-Key: abc-123-unique

Plain meaning

> "If this exact request is sent again, do NOT perform the action again."

Used in

- Payments
- Orders
- Refunds
- Any POST that causes side effects

Why it exists

- Network timeouts
- Client retries
- Page refreshes
- Double-clicks

Without idempotency → duplicate money / orders
With idempotency → safe retries

Correlation-ID / Trace-ID
X-Correlation-ID: order-98765

Purpose

- Track one request across:
    - API
    - Worker
    - Queue
    - External services

Tester value

- Log tracing
- Root cause analysis
- Faster debugging

Without correlation IDs, distributed systems are blind.

---

3 TIMEOUTS & RETRIES — THE SILENT KILLERS

What is a timeout?

Client waits for response.
No response within limit → client assumes failure.

Important:

- Timeout ≠ server failure
- Server may still be processing

What happens next?

Client retries the request.

If API is **not idempotent**:

- Same operation executes again
- Leads to **duplicate side effects**

This is how production incidents start.

---

4 STATUS CODES THAT ACTUALLY MATTER

200 OK

- Request processed
- Response returned

Used for:

- GET

- Some synchronous POSTs

Does NOT guarantee:

- Business success
- Final state

---

201 Created

- New resource created successfully
- Strong guarantee

Often includes:

Location: /orders/{id}

Used when:

- Creation is synchronous
- Resource definitely exists

---

202 Accepted (MOST IMPORTANT)

- Request accepted
- Processing will happen later
- Final result unknown at response time

Used for:

- Payments
- Refunds
- Async jobs

202 = “I got it, don’t ask me yet.”

---

5 INTERVIEW COMPARISON — 200 vs 202

Aspect	**200 OK**	**202 Accepted**
Processing	Completed	Pending
Business result	Known	Unknown
Async	Usually no	Yes
Safe to retry	Depends	Only with idempotency
Final truth	Yes	No

Interview line:

> “202 confirms acceptance, not business completion.”

---

6 REAL PRODUCTION FAILURE SCENARIO

Scenario

POST /payments
→ 202 Accepted

User refreshes page.
User is charged twice.

Step-by-step breakdown

1. Request sent
2. Server accepts → 202
3. Payment processing continues asynchronously
4. User refreshes page
5. Same request sent again
6. Server treats it as new
7. Payment processed again

Duplicate charge

Root Cause

- Async API
- Client retry
- **No Idempotency-Key**
- Server cannot identify duplicate request

---

7 CORRECT DESIGN FOR PAYMENT APIs

Correct request

```
POST /payments
Idempotency-Key: pay_123
```

Correct server behavior

- First request → process payment
- Retry with same key → return existing result
- No duplicate charge

---

8 TESTER CHECKLIST FOR ASYNC PAYMENT APIs

A real API tester verifies:

- With idempotency key
- Without idempotency key
- Retry same request
- Timeout + retry
- Page refresh
- Same key → same result
- Different key → new payment
- Final confirmation via webhook or status API

If you only test happy path, you miss real bugs.

---

9 WHY STRIPE / RAZORPAY DOCS MATTER

They document:

- Async behavior clearly

- Mandatory idempotency usage
- Retry safety
- Webhook-based confirmation
- Failure & compensation flows

Tester takeaway

- Response ≠ final truth
- Webhooks / polling confirm completion

KEY TAKEAWAYS (WRITE THIS IN BOLD IN GITHUB)

- HTTP is a **behavior contract**
- Headers matter more than body
- 202 means “wait and verify”
- Idempotency prevents production disasters
- Timeouts create retries
- Retries without idempotency create duplicates