FREE · LEARN.AIWITHPRIYABRAT.COM

The API Design Checklist

A printable, tick-box checklist for API design reviews and pull requests: naming, methods, errors, pagination, versioning, security, reliability, observability, LLM endpoints and launch.

By **Priyabrat Dalbehera** · GenAI engineer

2 pages | 14 areas | 78 checks to tick | 1 launch checklist

What's inside

1 **Design & contract** — naming, methods, bodies, status codes, errors, pagination, versioning — p. 2

2 **Security** — authentication, object-level authorisation, input safety, abuse — p. 2

3 **Operations** — reliability, performance, observability — p. 2

4 **LLM endpoints & launch** — streaming, budgets, guardrails, docs, tests, rollout — p. 2

How to use it
Go through each section before building an API, and again before it ships.

Unticked box?
Either fix it, or write down why it doesn't apply to this API.

Goes with
The REST API Design Cheat Sheet explains each rule with examples.

learn.aiwithpriyabrat.com

© 2026 Priyabrat Dalbehera · Free to download and share unchanged, with credit · Not for resale

API Design Checklist

14 areas, 78 checks · design · contract · security · reliability · performance · observability · LLM endpoints · launch

Tick before you build & ship

1 Naming and URLs

- ☐ Resources are nouns, plural: `/orders`, not `/getOrders`
- ☐ Lowercase with hyphens: `/order-items`
- ☐ IDs in the path, filters in the query: `/orders/42`, `?status=paid`
- ☐ Nesting at most one level: `/customers/7/orders`
- ☐ Verbs only for real actions: `POST /orders/42/cancel`
- ☐ One naming style for JSON fields everywhere (snake_case or camelCase)

Consistent names let people guess endpoints without reading the docs.

2 Methods

- ☐ `GET` never changes anything
- ☐ `POST` creates (returns 201 + `Location`) or runs an action
- ☐ `PUT` replaces the whole resource; `PATCH` changes some fields
- ☐ `DELETE` returns 204 and is safe to repeat
- ☐ Idempotent methods (GET, PUT, DELETE) really are idempotent

3 Requests and responses

- ☐ JSON in and out, with `Content-Type: application/json`
- ☐ Dates in ISO 8601 with a time zone: `2026-10-15T09:30:00Z`
- ☐ Money as integer minor units or a decimal string, never a float
- ☐ Enums documented; unknown values rejected
- ☐ Create and update return the full resource
- ☐ Optional fields are truly optional; nulls mean "no value", not "unknown"

4 Status codes

- ☐ 200 / 201 / 202 / 204 used for what they mean
- ☐ 400 malformed · 422 invalid values · 404 not found · 409 conflict
- ☐ 401 = not signed in · 403 = not allowed
- ☐ 429 with `Retry-After` when rate limited
- ☐ 500 only for real bugs, never for user mistakes
- ☐ Never `200` with `{"error": …}` in the body

5 Errors

- ☐ One error format for every endpoint (e.g. RFC 9457 problem details)
- ☐ Field-level messages for validation errors
- ☐ A `request_id` in every error, also in the logs
- ☐ No stack traces, SQL or internal hostnames in responses
- ☐ Machine-readable error codes plus human-readable messages

6 Lists, filters and pagination

- ☐ Every list endpoint is paginated, with a maximum `limit`
- ☐ Cursor pagination for large or fast-changing data
- ☐ Response says whether there's more: `next_cursor` / `has_more`
- ☐ Sorting with a stable tie-breaker (e.g. id)
- ☐ Filters documented and validated

7 Versioning and change

- ☐ A versioning plan from day one (`/v1/` or a version header)
- ☐ Adding fields is safe; removing or renaming needs a new version
- ☐ Clients are told to ignore unknown fields
- ☐ Deprecations announced with a date and a `Deprecation` header
- ☐ A changelog for API consumers

8 Authentication and authorisation

- ☐ HTTPS only; HSTS on
- ☐ Auth on every endpoint except a deliberate public list
- ☐ **Object-level checks**: this user may read or change *this* order
- ☐ User identity from the token, never from the request body
- ☐ Tokens short-lived; refresh tokens rotate
- ☐ API keys in headers, never in URLs or logs
- ☐ CORS allows only your own front-end origins

Watch out: broken object-level authorisation is the #1 API vulnerability (OWASP API Security Top 10).

9 Input safety and abuse

- ☐ Every input validated: type, range, length, format
- ☐ Request size limits; file uploads checked by type and size
- ☐ Parameterised queries everywhere (no SQL built from strings)
- ☐ Rate limits per API key or user, stricter on login and costly endpoints
- ☐ No mass assignment: only allowed fields are updatable

10 Reliability

- ☐ `Idempotency-Key` on POSTs that create payments or orders
- ☐ ETags + `If-Match` to stop lost updates
- ☐ Slow work runs in the background: 202 + job URL or a webhook
- ☐ Timeouts on every outgoing call; retries with backoff and jitter
- ☐ `/health` (alive) and `/ready` (dependencies OK) endpoints
- ☐ Webhooks signed (HMAC) and safe to receive twice

11 Performance

- ☐ No N+1 queries; related data loaded in one query
- ☐ Indexes for every filter and sort you offer
- ☐ Responses compressed (gzip / br)
- ☐ `Cache-Control` and ETags on cacheable reads
- ☐ Large exports streamed or generated as files, not one giant response

12 Observability

- ☐ Structured logs with request ID, user/key, route, status, latency
- ☐ No passwords, tokens or personal data in logs
- ☐ Metrics: request rate, error rate, latency p50/p95/p99 per endpoint
- ☐ Traces across services (OpenTelemetry)
- ☐ Alerts on error rate and latency, not only on downtime

13 LLM and AI endpoints

- ☐ Streaming responses (SSE) for generated text
- ☐ Long generations as jobs, not minutes-long requests
- ☐ Token-based limits and a budget per key
- ☐ Model name and prompt version logged with every call
- ☐ Input and output guardrails; retrieved text treated as untrusted
- ☐ Usage endpoint: tokens and cost per key or user

14 Docs, tests and launch

- ☐ OpenAPI spec with a summary, examples and error responses for every endpoint
- ☐ Contract tests run in CI against the spec
- ☐ Auth, permission and rate-limit tests, not only happy paths
- ☐ A sandbox or test keys for integrators
- ☐ Load test at 2–3× the expected peak
- ☐ Rollback plan and monitoring dashboards ready before launch

Tip: print this page and tick it off in your next design review or pull request.

learn.aiwithpriyabrat.com