FOLIO3 AI

PROMPT LIBRARY

120 ChatGPT Prompts for Software Development

Battle-tested prompt templates for scaffolding projects, writing production-ready APIs, debugging, testing, architecture design, DevOps, security, and more — copy, adapt, and ship faster.

120	25+	1
PROMPT TEMPLATES	DEV CATEGORIES	COPY & SHIP GUIDE

Folio3 AIML — Enterprise AI & Software Development
folio3.ai/prompts

Getting useful output from ChatGPT comes down to one thing: the prompt.

Vague inputs produce generic code, bloated explanations, and wasted back-and-forths. This guide cuts through that with 120 battle-tested ChatGPT prompts for software development, covering everything from debugging and code review to system design, documentation, and API integration. Copy them, adapt the bracketed placeholders to your stack, and ship faster.

What's inside

75 Write a deployment runbook
76 Build a tenant onboarding automation
77 Implement a GraphQL subscription
78 Design a disaster recovery plan
79 Build a changelog generator
80 Write a code generator
81 Implement dependency injection
82 Design a blue-green deployment strategy
83 Build a command query bus
84 Generate a Kubernetes deployment configuration
85 Implement API mocking for development
86 Build a GraphQL data loader
87 Design a configuration drift detection system
88 Implement a bulk API endpoint
89 Write a post-mortem report
90 Implement service mesh configuration
91 Build a developer feedback collection system
92 Implement a tiered storage system
93 Build a cross-service event schema registry
94 Implement graceful shutdown
95 Design a cost optimization strategy
96 Build a feature usage analytics system
97 Implement cross-origin resource sharing (CORS)
98 Build a dead letter queue handler
99 Implement a custom ORM layer
100 Build a zero-downtime database migration tool
101 Implement a content moderation system
102 Design a multi-region active-active architecture
103 Build a dynamic form builder backend
104 Implement a serverless function architecture
105 Generate a performance budget
106 Build a SaaS billing system integration
107 Implement a message queue consumer group
108 Design a read replica routing strategy
109 Build a request validation and sanitization layer
110 Implement a saga pattern for distributed transactions
111 Write a database connection pooling configuration
112 Build a client-side state management architecture
113 Implement API request signing
114 Design a multi-step checkout flow backend
115 Build a code coverage enforcement pipeline
116 Implement an outbox pattern for reliable event publishing
117 Write a code complexity analysis report
118 Build a request context propagation system
119 Design an API deprecation and sunset process
120 Implement a rollback-safe blue/green database schema strategy

01

PROJECT SETUP

Set up a new project scaffold

```
Act as a senior software engineer. I want to start a new [web / mobile / backend /
CLI] project using [tech stack, e.g., React + Node.js / Django / Spring Boot].

Generate a complete project scaffold that includes:

Recommended folder and file structure with explanation for each directory
Essential configuration files (package.json / requirements.txt / pom.xml / .env.example)
Linting and formatting setup (ESLint + Prettier / Black + isort / Checkstyle)
Git configuration: .gitignore, commit message convention, and branch naming strategy
README template with sections: overview, prerequisites, setup, run, test, deploy
A Makefile or task runner config with commands for build, test, lint, and run
Prioritize developer experience and team scalability. Explain any non-obvious choices.
```

02

BACKEND DEVELOPMENT

Write a clean REST API endpoint

```
You are a backend engineer. I need to implement a REST API endpoint for [describe the
resource and action, e.g., "creating a new user account"].

Build a production-ready endpoint that includes:

Route definition with correct HTTP method and URL path following REST conventions
Input validation with descriptive error messages for each field
Business logic layer separated from the controller
Database interaction using [ORM / raw SQL / query builder] — write parameterized queries only
Proper HTTP status codes for success (201 / 200) and all failure paths (400, 401, 403, 404, 500)
Consistent JSON response format: { success, data, message, errors }
Unit tests covering happy path, validation failure, and database error scenarios
Use [language/framework]. Follow SOLID principles and include inline comments for non-obvious logic.
```

03

DATABASE DESIGN

Design a normalized database schema

```
Act as a database architect. I am building [describe the application, e.g., "an
e-commerce platform with products, orders, and customers"].

Design a normalized relational schema that:

Identifies all entities and their attributes
Defines primary keys, foreign keys, and junction tables for many-to-many relationships
Applies normalization up to 3NF — explain any deliberate denormalization decisions
Specifies data types, constraints (NOT NULL, UNIQUE, CHECK), and default values
Adds indexes for all foreign keys and expected high-frequency query columns
Includes an ER diagram description (or ASCII diagram) showing all relationships
Provides the full DDL SQL to create all tables
Use [PostgreSQL / MySQL / SQLite]. Anticipate [expected scale, e.g., 1M rows/month] and call out any
scalability considerations.
```

04

DEBUGGING

Debug a failing function

```
Act as a senior debugger. The following [language] function is producing incorrect
output or throwing an error:

[paste your function here]
Error or unexpected behavior: [describe what's happening vs. what's expected]
Input I'm testing with: [provide sample inputs]

Do the following:

Identify the root cause — not just the symptom
Explain why the bug occurs in plain English
Show the corrected function with the fix highlighted
Add at least 3 test cases (edge cases included) that would catch this bug going forward
List any other code smells or latent bugs you spot in the function
Suggest a refactor if the logic can be simplified
Do not guess — trace the execution step by step before concluding.
```

05

TESTING

Write comprehensive unit tests

```
You are a QA engineer and testing expert. I have the following function / class /
module:

[paste your code here]
Write a full unit test suite in [Jest / PyTest / JUnit / RSpec] that:

Tests every public method with at least one happy path test
Covers all branching logic (if/else, switch, early returns)
Tests boundary conditions: empty input, null/undefined, zero, max value, special characters
Mocks all external dependencies (DB calls, API calls, file I/O) correctly
Includes at least 2 negative tests for expected error/exception scenarios
Uses descriptive test names that read as sentences: "should return X when Y"
Groups tests logically using describe/context blocks
Aim for 90%+ branch coverage. Explain any test you consider the most important and why.
```

06

CODE QUALITY

Refactor legacy code

```
Act as a software architect specializing in code quality. Here is a legacy [language]
code block that needs to be modernized:

[paste your legacy code here]
Context: This code was written [X years ago] and runs in production. It has [known problems: no
tests /
performance issues / hard to read / tightly coupled].

Refactor it to:

Apply [SOLID / DRY / KISS / YAGNI] principles — call out which ones you're applying and where
Replace anti-patterns with idiomatic [language] patterns
Break large functions into single-responsibility units
Improve naming for all variables, functions, and parameters
Add JSDoc / docstrings to all public functions
Preserve 100% behavioral parity — list any assumptions you make
Show a before/after diff-style comparison of the most impactful changes
Do not introduce new dependencies unless clearly justified.
```

07

SECURITY

Implement authentication and authorization

```
You are a security-focused backend engineer. I need to implement auth for a [REST API
/ GraphQL API / web app] using [JWT / OAuth2 / session-based auth].

Build a complete auth system that includes:

User registration with password hashing using bcrypt (cost factor ≥ 12)
Login endpoint that returns [access token + refresh token / session cookie]
JWT signing with RS256 (preferred) or HS256 — explain the tradeoff
Access token expiry of 15 minutes; refresh token rotation with 7-day expiry
Middleware to protect routes: verify token, extract user identity, attach to request context
Role-based access control (RBAC) with at least 3 roles: admin, editor, viewer
Rate limiting on login endpoint to prevent brute force
Secure cookie flags (httpOnly, Secure, SameSite=Strict) if using cookies
Use [language/framework]. Follow OWASP top 10 guidelines and call out any security decisions
explicitly.
```

08

PERFORMANCE

Optimize a slow database query

```
Act as a database performance engineer. The following SQL query is running slowly on
a table with [X million rows]:

sql
[paste your query here]
Current execution time: [X seconds]
Table schema: [describe relevant columns and existing indexes]
Database: [PostgreSQL / MySQL / SQL Server]

Do the following:

Run an EXPLAIN ANALYZE interpretation — describe what each step costs
Identify the performance bottlenecks: missing indexes, full table scans, N+1 patterns, inefficient
joins
Rewrite the query with optimizations applied
Specify which indexes to create (type, columns, partial/composite) and why
Estimate the performance improvement expected
List 3 query patterns in this codebase I should audit for similar issues
Show the original vs. optimized query side by side and explain every change made.
```

09

DEVOPS

Build a CI/CD pipeline

```
You are a DevOps engineer. I want to set up a CI/CD pipeline for a
[language/framework] application hosted on [GitHub / GitLab / Bitbucket] and deploying to [AWS / GCP /
Azure
/ Kubernetes / Vercel].

Build a complete pipeline configuration that:

Triggers on: PR open (CI only), merge to main (CI + CD), and manual deployment to production
CI stage: install dependencies, lint, run unit tests, run integration tests, build artifact
CD stage: build and push Docker image to [ECR / GCR / DockerHub], deploy to [environment]
Uses environment-specific variables for staging vs. production (never hardcoded)
Implements rollback: automatic on health check failure, manual trigger available
Sends deployment notifications to [Slack / Teams / email]
Caches dependencies between runs to minimize build time
Output the full YAML configuration with inline comments. Flag any secrets that must be configured in
the
CI/CD platform.
```

10

DEVOPS

Create a Docker containerization setup

```
Act as a containerization expert. I have a [language/framework] application that I
need to containerize for [development / staging / production].

Create a complete Docker setup including:

A multi-stage Dockerfile: build stage + lean production image (use Alpine or Distroless where
possible)
.dockerignore file — explain why each entry matters
docker-compose.yml for local development with all services: app, database, cache, optional message
queue
Health check configuration for each service
Volume mounts for development hot-reload and persistent data
Non-root user setup in the container for security
Environment variable handling using .env file with .env.example template
Optimize for image size and startup time. Explain any layer caching strategies applied.
```

11

ARCHITECTURE

Design a microservices architecture

```
You are a solutions architect. I want to decompose a monolithic [describe your app]
into microservices.

The monolith currently handles: [list main features / modules]
Expected scale: [concurrent users, requests/second, data volume]
Team size: [number of developers / teams]

Design a microservices architecture that:

Identifies service boundaries using domain-driven design (DDD) — define each bounded context
Specifies the API contract between services (REST / gRPC / events)
Recommends synchronous vs. asynchronous communication per service pair and explains why
Designs the event-driven backbone if applicable (Kafka / RabbitMQ / SQS) — topic/queue design
Addresses cross-cutting concerns: auth, logging, tracing, rate limiting (API gateway or sidecar)
Defines a data ownership model — no shared databases
Provides a phased migration plan from monolith to microservices with zero downtime
Include an architecture diagram description with all services, communication patterns, and data stores
labeled.
```

12

API DEVELOPMENT

Write a GraphQL schema and resolvers

```
Act as a GraphQL API expert. I need to build a GraphQL API for [describe your domain,
e.g., a blog platform with posts, authors, and comments].

Deliver a complete implementation that includes:

Full SDL schema: all types, queries, mutations, and subscriptions with descriptions
Input types with validation rules defined in schema
Resolver implementations for each query and mutation using [DataLoader / batch loading] to avoid N+1
queries
Authentication directive (@auth) and authorization logic in resolvers
Error handling using union types (Result | Error pattern) — not generic errors
Pagination on all list queries using cursor-based pagination (not offset)
Example queries and mutations for each operation with expected response shapes
Use [Apollo Server / Nexus / Pothos / Strawberry]. Include performance considerations for deeply
nested
queries.
```

13

PERFORMANCE

Implement a caching strategy

```
You are a backend performance engineer. My [API / application] is experiencing high
latency due to expensive operations: [describe: DB queries, third-party API calls, heavy computation].

Design and implement a multi-layer caching strategy that:

Identifies what to cache vs. what must always be fresh — provide decision criteria
Implements in-memory caching (LRU cache) for [specific use case]
Implements distributed caching with Redis: key design, TTL strategy, and eviction policy
Handles cache invalidation: TTL-based, event-driven, and manual purge endpoints
Adds cache-aside vs. write-through vs. write-behind — recommend the right pattern per data type
Prevents cache stampede using probabilistic early expiration or locking
Adds cache hit/miss metrics instrumentation
Show complete code for the caching layer and explain the TTL values chosen for each data type.
```

14

REAL-TIME

Build a real-time WebSocket feature

```
Act as a real-time systems engineer. I want to add [live chat / live notifications /
collaborative editing / real-time dashboard updates] to my [framework] application.

Build a production-ready WebSocket implementation that:

Sets up WebSocket server with connection lifecycle management (connect, message, disconnect, error)
Implements rooms/channels so messages are scoped to the right users
Handles authentication of WebSocket connections using the existing JWT-based auth
Broadcasts events to specific users, rooms, or all connected clients as needed
Implements heartbeat/ping-pong to detect and clean up stale connections
Adds reconnection logic on the client side with exponential backoff
Handles horizontal scaling — how to share state across multiple server instances using Redis Pub/Sub
Use [Socket.io / ws / native WebSocket]. Include both server and client-side code.
```

15

DOCUMENTATION

Generate API documentation

```
You are a technical writer and API expert. I have the following API with these
endpoints:

[paste your route definitions, controller signatures, or OpenAPI partial here]
Generate complete API documentation that includes:

OpenAPI 3.1 YAML spec for all endpoints with full request/response schemas
Description for every endpoint: purpose, who should use it, and any rate limits
All request parameters: path, query, header, and body — with type, required/optional, and example
values
All possible response codes with schema and plain-English explanation of when each occurs
Authentication section: how to obtain and pass credentials
At least 2 curl examples per endpoint (success and a common failure)
A Postman collection JSON ready to import
Format the OpenAPI spec to be importable into Swagger UI, Redoc, or Postman without modification.
```

16

BACKEND DEVELOPMENT

Implement error handling middleware

```
Act as a senior backend engineer. My [framework] application has inconsistent error
handling — some errors crash the server, others return unhelpful messages, and none are logged
consistently.

Build a centralized error handling system that:

Defines a custom error class hierarchy: AppError → ValidationError, AuthError, NotFoundError,
DatabaseError
Includes error code, HTTP status, user-facing message, and internal debug message per error type
Implements global error handling middleware that catches all unhandled errors
Returns consistent JSON error response: { success: false, error: { code, message, details } }
Sanitizes error messages — never expose stack traces or DB internals to clients
Logs all errors with: timestamp, request ID, user ID, route, error type, stack trace
Sends alerts for 5xx errors via [Slack webhook / PagerDuty / email]
Show the full implementation and demonstrate how to throw and catch errors across controllers,
services, and
middleware.
```

17

DATABASE

Write a database migration

```
You are a database engineer. I need to make the following schema change to a
production database with [X million rows] and zero-downtime requirement:

Change required: [describe exactly what needs to change — add column, rename column, change type, add
index,
split table]

Current schema: [paste relevant DDL]

Deliver a safe migration plan that:

Uses expand-contract (parallel change) pattern to ensure backward compatibility
Writes the migration script with UP and DOWN operations
Adds the new column as nullable first, then backfills data in batches of [1000–10000] rows to avoid
locking
Creates indexes concurrently (CONCURRENTLY keyword for PostgreSQL)
Handles rollback: what to do if the migration fails halfway
Includes a pre-migration checklist (backups, disk space, replication lag check)
Provides estimated migration time based on table size
Use [Flyway / Alembic / Liquibase / raw SQL] migration tooling.
```

18

BACKEND DEVELOPMENT

Implement a job queue and background worker

```
Act as a backend systems engineer. I need to offload [describe the heavy task: email
sending, image processing, report generation, third-party API calls] to a background job queue.

Build a complete job queue system that:

Sets up [BullMQ / Celery / Sidekiq / Hangfire] with [Redis / RabbitMQ] as the broker
Defines job schema with all required payload fields and their types
Implements the worker with: job processing logic, success handler, and failure handler
Configures retry strategy: max 3 attempts with exponential backoff (1s, 5s, 30s)
Handles job timeout and marks stuck jobs as failed after [X minutes]
Adds job progress tracking and status polling endpoint for the client
Implements dead letter queue for permanently failed jobs with alerting
Include the producer (enqueue) and consumer (process) code, plus a monitoring dashboard setup if the
library
supports it.
```

19

SECURITY & PERFORMANCE

Design a rate limiting system

```
You are a backend engineer. I need to implement rate limiting on my [REST API /
GraphQL API] to prevent abuse, protect upstream services, and ensure fair usage.

Build a rate limiting system that:

Implements sliding window counter algorithm using Redis (not fixed window — explain why)
Applies limits at multiple levels: per IP, per authenticated user, and per API key
Sets different limits per route tier: public endpoints (100 req/min), authenticated (1000 req/min),
premium
(10000 req/min)
Returns proper HTTP 429 response with Retry-After and X-RateLimit-* headers
Whitelists internal services and health check endpoints
Logs rate limit violations with user/IP for abuse detection
Handles distributed rate limiting correctly across multiple server instances
Show the middleware code, Redis key design, and the response headers format.
```

20

BACKEND DEVELOPMENT

Implement file upload and storage

```
Act as a full-stack engineer. I need to implement file upload functionality for
[images / documents / videos / any file type] in my [framework] application.

Build a complete file handling system that:

Validates file on upload: type whitelist (MIME + extension), max size [X MB], and virus scan hook
Streams large files directly to [S3 / GCS / Azure Blob] without loading into memory
Generates a unique, non-guessable key for each file (UUID + original name sanitized)
Creates signed pre-signed URLs for secure client-side uploads (bypassing the server for large files)
Generates thumbnails or previews for [images / PDFs] using [Sharp / Pillow / ImageMagick]
Stores file metadata in the database: original name, size, MIME type, storage key, uploader ID
Implements soft delete — marks files as deleted in DB, runs a scheduled cleanup job for actual
deletion
Include both the backend API and the frontend upload component with progress indicator.
```

21

FEATURE DEVELOPMENT

Build a search feature with full-text search

```
You are a search engineer. I want to add full-text search to my application for
[describe what users are searching: products, blog posts, users, documents].

Implement a complete search system that:

Evaluates [PostgreSQL full-text search / Elasticsearch / Meilisearch / Typesense] for my scale and
chooses
the best fit — justify the recommendation
Sets up the search index with the right field mappings and analyzers for [language]
Implements the search query with: fuzzy matching, typo tolerance, phrase search, and field boosting
Adds filters: faceted search by [category, price range, date] with facet counts
Returns results ranked by relevance with highlighted matched terms
Implements autocomplete / search-as-you-type with debouncing on the frontend
Handles index synchronization: update index on create, update, and delete in the DB
Show indexing code, search query, and the frontend search component.
```

22

TOOLING

Write a CLI tool

```
Act as a developer tooling engineer. I want to build a command-line interface (CLI)
tool that [describe what the tool does, e.g., "scaffolds new microservices, manages environment
configs, and interacts with our internal API"].

Build a production-quality CLI that:

Uses [Commander.js / Click / Cobra / Picocli] for command parsing — justify your choice
Defines at least 3 subcommands with flags, options, and arguments clearly structured
Validates all input with helpful, color-coded error messages
Implements interactive prompts for missing required inputs (using Inquirer / Click prompts)
Reads configuration from [~/.config/mytool/config.yml] and supports override via env vars
Shows progress spinners for long-running operations and a success/failure summary at the end
Includes --version, --help on all commands, and man page-style usage examples
Package it as an npm package / pip package / single binary. Include installation instructions.
```

23

DEVOPS & OBSERVABILITY

Implement logging and observability

```
You are an SRE engineer. My application has no structured logging or observability. I
need to instrument it properly for production.

Set up a complete observability stack that:

Implements structured JSON logging with: timestamp, log level, request ID, user ID, service name,
message,
and metadata
Adds request ID propagation across all services and log entries using middleware
Integrates distributed tracing with [OpenTelemetry + Jaeger / Zipkin / DataDog]
Instruments all DB queries, external API calls, and cache operations with spans
Defines key metrics: request rate, error rate, p50/p95/p99 latency, queue depth — expose via
[Prometheus
endpoint / StatsD]
Creates a Grafana dashboard JSON with the 5 most important panels for the on-call engineer
Sets up alerting rules: error rate > 1%, p99 latency > 2s, service down > 30s
Show the logging middleware, tracing setup, and metric instrumentation code.
```

24

ARCHITECTURE

Design a data access layer (DAL)

```
Act as a software architect. I need to build a clean data access layer for my
[language/framework] application using [PostgreSQL / MySQL / MongoDB / DynamoDB].

Design and implement a DAL that:

Defines repository interfaces: one per aggregate root / entity (e.g., UserRepository, OrderRepository)
Implements concrete repository classes with full CRUD + domain-specific query methods
Handles database connection pooling configuration (min, max, timeout, idle timeout)
Wraps all DB errors in domain-specific errors (not raw DB exceptions in the application layer)
Implements the Unit of Work pattern for transactions spanning multiple repositories
Uses query builders or typed ORMs — avoids raw string SQL in application code
Makes the DAL easily swappable: program to the interface, inject the implementation
Show the interface, implementation, and a service class consuming the DAL correctly.
```

25

AUTHENTICATION

Build an OAuth2 integration

```
You are an authentication engineer. I want to add "Sign in with [Google / GitHub
/ Microsoft / Apple]" to my [web / mobile] application.

Build a complete OAuth2 integration that:

Implements the Authorization Code Flow with PKCE (required for SPAs and mobile)
Builds the authorization URL with correct scopes, state parameter (CSRF protection), and nonce
Handles the callback: validates state, exchanges code for tokens, and handles errors from the provider
Fetches the user profile from the provider's API and maps it to your internal user model
Links OAuth accounts to existing email-based accounts (account merging logic)
Stores refresh tokens securely (encrypted at rest) and refreshes access tokens transparently
Handles token revocation on logout and account disconnect
Show server-side code for all endpoints plus the frontend redirect initiation. Include error handling
for
all OAuth failure modes.
```

26

FRONTEND & BACKEND

Implement pagination and infinite scroll

```
Act as a full-stack engineer. I need to implement pagination on an endpoint that
returns [describe the resource] with [estimated row count].

Build a complete pagination system that:

Evaluates cursor-based vs. offset-based pagination for my use case — recommend the right approach
Implements the backend endpoint with: page size defaulting to 20 (max 100), cursor or page parameter,
and
sort options
Returns pagination metadata: { data, total, nextCursor, hasNextPage } or { data, page, totalPages }
Implements stable sort (no skipped or duplicated rows on concurrent inserts)
Builds the frontend component with: loading state, empty state, error state, and end-of-list state
Implements infinite scroll using IntersectionObserver (not scroll events) with prefetch trigger at 80%
scroll
Handles deep linking — the URL reflects the current page or cursor so users can share/bookmark
Show both the API and the frontend component code.
```

27

DEVELOPER EXPERIENCE

Write a data seeder for development

```
You are a developer experience engineer. I need to populate a development database
with realistic test data for [describe your schema: tables/collections and their relationships].

Build a comprehensive data seeder that:

Generates [X] rows per table in the correct order to satisfy foreign key constraints
Uses [Faker.js / Faker (Python) / Bogus] to generate realistic field values per data type
Creates deterministic data with a fixed seed so all developers get the same dataset
Handles many-to-many relationships and junction tables correctly
Avoids duplicate key violations on re-runs: truncate-then-seed or upsert strategy
Creates specific named test fixtures: admin user, free user, premium user, banned user — for use in
tests
Is runnable as a single command: make seed or npm run db:seed
Output the full seeder script. Include a summary of what gets created when it runs.
```

28

ENGINEERING PRACTICES

Implement feature flags

```
Act as a platform engineer. I want to implement feature flags in my application to
support [gradual rollouts / A/B testing / kill switches / beta user access].

Build a feature flag system that:

Defines flags in a config file or admin UI with: flag key, description, enabled state, and rollout
percentage
Supports targeting rules: enable for specific user IDs, email domains, or user attributes
Evaluates flags at request time with fallback to default if the flag service is unavailable
Implements a lightweight SDK that checks flags in-memory (loaded at startup, refreshed every 60s)
Adds flag evaluation to middleware and makes the result available throughout the request context
Logs all flag evaluations for auditability and to debug unexpected behavior
Integrates with [LaunchDarkly / Unleash / Flagsmith / self-hosted Redis store] — or builds a simple
version
from scratch
Show the flag evaluation logic, targeting rule engine, and an example of guarding a new feature with a
flag.
```

29

CODE QUALITY

Conduct a code review

```
You are a senior engineer conducting a formal code review. Here is the code submitted
in a pull request:

[paste the code to review]
Context: This code [adds a new feature / fixes a bug / refactors existing logic] in a [production /
internal
/ open-source] codebase.

Review the code and provide structured feedback covering:

Correctness: Are there bugs, edge cases not handled, or incorrect logic?
Security: Any SQL injection, XSS, insecure deserialization, hardcoded secrets, or OWASP concerns?
Performance: Any N+1 queries, memory leaks, unbounded loops, or missing indexes?
Maintainability: Is the code readable, well-named, and appropriately commented?
Test coverage: What is missing from tests? Are mocks used correctly?
Architecture: Does it follow existing patterns? Any violations of SOLID or DRY?
Blocking vs. non-blocking feedback: Label each comment as [MUST FIX] / [SUGGESTION] / [NITPICK]
End with an overall verdict: Approve / Request Changes / Needs Discussion.
```

30

REAL-TIME

Build a WebSocket-based collaborative editing feature

```
Act as a real-time collaboration engineer. I want to add collaborative document
editing [like Google Docs] to my application where multiple users can edit the same document
simultaneously.

Implement a conflict-free collaborative editing system that:

Explains the choice between OT (Operational Transformation) and CRDT — recommend the right approach
for my
use case
Implements the chosen algorithm for handling concurrent edits without conflicts
Broadcasts operations via WebSocket to all connected users in real time
Shows each user's cursor position in the document with their name and a unique color
Handles reconnection: replays missed operations since last known state when a user reconnects
Persists the document state periodically (every 30s or on disconnect) to the database
Implements presence awareness: who is currently viewing/editing the document
Use [Yjs / Automerge / ShareDB] and show both server and client implementation.
```

31

PERFORMANCE

Generate a performance profiling report

```
You are a performance engineering expert. I have a [language] application that is
slow under load. I ran the following profiler output:

[paste profiler results, flame graph description, or slow query log]
Analyze the profiling data and provide:

The top 3 bottlenecks ranked by impact (% of total time / resources consumed)
Root cause analysis for each bottleneck — not just "this function is slow" but why
A concrete fix for each bottleneck with expected improvement estimate
Code changes needed to implement each fix
How to verify the fix worked — which metric to measure and by how much it should improve
Recommended profiling tools and cadence to run going forward
A list of 5 proactive performance practices to prevent future regressions
Be specific — reference function names, line numbers, and actual numbers from the profiler output.
```

32

PERFORMANCE TESTING

Write a load testing script

```
Act as a performance test engineer. I need to load test my [API / service] to
validate it can handle [X concurrent users / Y requests per second] before going to production.

Create a complete load test suite using [k6 / Locust / Artillery / JMeter] that:

Defines realistic user scenarios based on actual usage patterns: [describe the key flows]
Implements [X] virtual users ramping up over [Y minutes] to the target load
Includes think time between requests to simulate real user behavior
Parameterizes test data (user credentials, search terms, product IDs) from a CSV or inline list
Sets pass/fail thresholds: p95 latency < 500ms, error rate < 0.1%, throughput ≥ [X req/s]
Produces a results report: throughput, response time distribution, error log, and bottleneck
identification
Includes a stress test scenario that ramps to 200% of target to find the breaking point
Output the full test script plus instructions on how to run it locally and in CI.
```

33

DEVELOPER TOOLING

Create an API client SDK

```
You are an SDK engineer. I want to create a client SDK for my REST API in [JavaScript
/ Python / Go / Java] so that developers can integrate with it easily.

Build a production-grade SDK that:

Auto-generates typed models for all request and response schemas
Implements retry logic: exponential backoff with jitter for 429 and 5xx responses (max 3 retries)
Handles authentication transparently: accepts API key or OAuth token, refreshes tokens automatically
Provides both promise-based and async/await interfaces (for JS/Python)
Implements request timeouts with sensible defaults (30s connect, 60s read)
Includes comprehensive error types: ApiError, AuthenticationError, RateLimitError, NetworkError
Publishes to [npm / PyPI / pkg.go.dev] with a README showing the top 5 use cases and a quickstart
Show the core HTTP client, a sample resource class (e.g., UsersClient), and end-to-end usage examples.
```

34

ARCHITECTURE

Implement event sourcing

```
Act as a software architect. I want to implement event sourcing for the [describe
your domain, e.g., order management / account transactions / inventory] module of my application.

Design and implement a complete event sourcing system that:

Defines the domain events as immutable value objects with: eventType, aggregateId, version, timestamp,
payload
Implements the event store: append-only writes, read by aggregateId, and optimistic concurrency with
version
checking
Builds the aggregate: loads state by replaying all events, applies new events, and emits them to the
store
Implements snapshots for aggregates with >100 events to avoid slow replay
Projects events into read models (query-side) using [PostgreSQL / Redis / Elasticsearch]
Handles event versioning: how to evolve events without breaking existing data (upcasting pattern)
Provides a complete example: create order → add item → payment confirmed → order shipped — end to end
Use [EventStoreDB / custom implementation]. Show the event store, aggregate, and one projection.
```

35

ARCHITECTURE

Implement the CQRS pattern

```
You are a solutions architect. I want to apply the Command Query Responsibility
Segregation (CQRS) pattern to my [describe your domain] to improve scalability and separation of
concerns.

Design a CQRS implementation that:

Separates the write model (commands) and read model (queries) at the code and data level
Defines commands as intent objects: CreateOrder, CancelOrder, UpdateShippingAddress
Implements command handlers that validate, execute business logic, and persist to the write store
Defines queries as specific read request objects: GetOrderSummary, ListOrdersByCustomer
Implements query handlers that read from optimized read models (denormalized for fast reads)
Propagates changes from write to read model: synchronously (same transaction) or asynchronously
(events)
Explains when CQRS adds value vs. when it adds unnecessary complexity — and the tradeoffs for my case
Show a complete end-to-end example from HTTP request → command → handler → read model update → query.
```

36

ARCHITECTURE

Build a multi-tenant SaaS architecture

```
Act as a SaaS architect. I am building a multi-tenant application for [describe the
product] and need to decide on a tenancy model and implement tenant isolation.

Design a multi-tenant architecture that:

Evaluates the 3 tenancy models (shared DB, schema-per-tenant, DB-per-tenant) and recommends the right
approach based on [expected tenant count, compliance requirements, customization needs]
Implements tenant identification: subdomain-based, JWT claim-based, or API key header
Adds tenant context middleware that extracts tenant ID and attaches it to every request
Ensures all DB queries are scoped to the current tenant — no cross-tenant data leakage
Implements tenant provisioning: creates the tenant record, provisions their data space, and seeds
default
data
Handles tenant-specific configuration: feature flags, branding, limits
Designs the billing model integration: how usage is tracked per tenant for metering
Show the tenant middleware, query scoping strategy, and a sample onboarding flow.
```

37

DEVOPS

Write infrastructure as code (IaC)

```
You are a DevOps engineer. I need to provision the infrastructure for a [describe the
system: web app, API, data pipeline] on [AWS / GCP / Azure].

Required infrastructure:

Compute: [EC2 / ECS / EKS / Cloud Run / Lambda]
Database: [RDS PostgreSQL / Cloud SQL / Aurora / DynamoDB]
Cache: [ElastiCache Redis / Memorystore]
CDN: [CloudFront / Cloud CDN]
Networking: [VPC, subnets, security groups, NAT gateway]
Write Terraform (or Pulumi) configuration that:

Provisions all resources with least-privilege IAM roles
Separates environments using [workspaces / modules]: dev, staging, production
Stores Terraform state in [S3 + DynamoDB locking / GCS]
Uses variables for all environment-specific values — no hardcoded ARNs or IDs
Includes resource tagging: environment, project, owner, cost center
Outputs all critical values: endpoint URLs, ARNs, connection strings (use Secrets Manager for
sensitive
ones)
Provide the full .tf files and a README on how to init, plan, and apply.
```

38

BACKEND DEVELOPMENT

Design a webhook system

```
Act as a backend engineer. I want to implement a webhook system so that my platform
can notify external systems when [describe events: order placed, payment completed, user registered].

Build a production-ready webhook system that:

Allows customers to register webhook endpoints via API: URL, secret, and event type subscriptions
Delivers webhooks asynchronously via a job queue — never in the request path
Signs each payload using HMAC-SHA256 with the customer's secret and sends the signature in a header
Retries failed deliveries: max 5 attempts with exponential backoff (1m, 5m, 30m, 2h, 24h)
Records every delivery attempt: timestamp, HTTP status, response body, and latency
Provides a dashboard endpoint for customers to view delivery history and manually retry failed events
Implements a test mode: customers can send a test event to verify their endpoint is working
Show the registration API, delivery worker, signature verification guide for customers, and the retry
logic.
```

39

API DESIGN

Implement API versioning

```
You are an API design expert. My REST API has been in production for [X months] and I
need to introduce breaking changes while keeping existing clients working.

Design and implement an API versioning strategy that:

Evaluates URL versioning (/v1/), header versioning (Accept: application/vnd.api+json;version=2), and
query
param versioning — recommend the best fit and justify
Implements the chosen strategy with routing that maps versions to the right controller/handler
Defines a deprecation policy: minimum [X months] notice, deprecation header in responses, sunset date
Handles shared code between versions: how to avoid duplicating business logic
Implements a version negotiation middleware that selects the right version or returns 400 for
unsupported
versions
Documents the migration guide from v1 to v2 for each breaking change
Adds version metrics: track usage per version to know when it's safe to retire an old version
Show the routing config, middleware, and a concrete example of migrating one endpoint from v1 to v2.
```

40

FEATURE DEVELOPMENT

Build a notification system

```
Act as a backend engineer. I need to build a multi-channel notification system for my
application that sends [email, push, SMS, in-app] notifications.

Design a notification system that:

Defines a unified notification interface with: recipient, channel, template ID, variables, and
priority
Routes notifications to the right channel based on user preferences and notification type
Integrates with: [SendGrid / SES] for email, [FCM / APNs] for push, [Twilio] for SMS
Implements template rendering with variable substitution and HTML/plain text variants for email
Queues all notifications via a job queue with channel-specific workers
Handles delivery failures: retry for transient errors, fallback channel if primary fails
Tracks delivery status per notification: sent, delivered, opened, clicked, failed
Show the notification service interface, the email worker, and the user preference API.
```

41

SECURITY

Design a secrets management strategy

```
You are a security engineer. My application currently has secrets [hardcoded in
config files / in environment variables in .env files / committed to the repo] and needs to be secured
properly.

Implement a proper secrets management strategy that:

Audits and catalogs all current secrets: DB passwords, API keys, JWT signing keys, OAuth secrets
Migrates all secrets to [HashiCorp Vault / AWS Secrets Manager / GCP Secret Manager / Azure Key Vault]
Implements dynamic secrets for database credentials: short-lived credentials generated per service
Rotates all secrets without downtime: blue-green rotation strategy for each secret type
Injects secrets into the application at runtime (not baked into images or config files)
Implements least-privilege access: each service can only read the specific secrets it needs
Adds secret access audit logging and alerts for unauthorized access attempts
Show the Vault/Secrets Manager setup, the injection mechanism, and the rotation script for one
credential
type.
```

42

SECURITY

Implement data encryption

```
Act as a security engineer. I need to encrypt sensitive data in my application: [list
sensitive fields: SSNs, credit card numbers, health records, PII].

Design a field-level encryption system that:

Chooses the right encryption approach: AES-256-GCM for data at rest, TLS for data in transit — explain
when
each applies
Implements transparent field-level encryption: encrypts before write, decrypts after read in the data
layer
Manages encryption keys using [KMS / HSM / Vault] — never stores keys alongside data
Implements key rotation: re-encrypts data with new key without downtime
Uses envelope encryption: data encrypted with a data key, data key encrypted with a master key
Handles searchable encryption where needed (blind indexing or order-preserving encryption with
tradeoff
explanation)
Adds tamper detection using MAC or digital signatures on sensitive records
Show the encryption/decryption utilities, key management calls, and the repository layer implementing
transparent encryption.
```

43

ARCHITECTURE

Build a plugin/extension architecture

```
You are a software architect. I want to make my [application type] extensible via a
plugin system so that [internal teams / third-party developers] can add new capabilities without
modifying
core code.

Design a plugin architecture that:

Defines the plugin interface: required methods (initialize, teardown), lifecycle hooks, and metadata
schema
Implements a plugin registry: discovers, loads, and validates plugins at startup
Uses dependency injection to provide plugins with sandboxed access to core services
Implements an event/hook system: plugins can subscribe to and emit named events
Enforces plugin isolation: a crashing plugin should not bring down the core application
Handles plugin versioning and compatibility checking against the core API version
Provides a starter plugin template and developer documentation
Show the plugin interface, the registry implementation, an example plugin, and how the core
application
calls plugins at lifecycle events.
```

44

FEATURE DEVELOPMENT

Implement a state machine

```
Act as a software engineer. I have a domain entity that has complex state
transitions: [describe the entity and its states, e.g., "an order that can be: draft → submitted →
approved → shipped → delivered → cancelled"].

Implement a formal state machine that:

Defines all states as an enum or constants
Defines all valid transitions as a transition table: { from, event, to, guard, action }
Implements guard conditions that prevent invalid transitions (e.g., can't ship an unpaid order)
Executes side effects (actions) on transitions: send notification, update audit log, call external API
Throws a clear error for any attempted invalid transition
Persists the current state to the database and records all transitions in a state history table
Provides a method to visualize the state machine as a diagram (Mermaid or Graphviz notation)
Use [XState / python-statemachine / Spring State Machine / custom implementation]. Show the full
machine
definition and a usage example.
```

45

API DESIGN

Design an idempotent API

```
You are an API design expert. I need my [payment processing / order creation /
resource provisioning] API endpoints to be idempotent so that duplicate requests (retries, network
failures)
don't cause duplicate operations.

Implement idempotency for my API that:

Accepts an Idempotency-Key header on all mutating endpoints (POST, PATCH)
Stores the idempotency key + response in a cache (Redis) with a 24-hour TTL
On duplicate request: returns the cached response without re-executing the business logic
Handles concurrent duplicate requests: uses Redis locking to prevent race conditions
Scopes idempotency keys per user or API key — keys are not global
Returns the same HTTP status and response body for both the original and duplicate requests
Documents the idempotency behavior in the API docs with retry guidance for clients
Show the idempotency middleware, the Redis storage pattern, and a test demonstrating the behavior.
```

46

BACKEND DEVELOPMENT

Build a scheduled jobs system

```
Act as a backend engineer. I need to run scheduled background jobs: [list your jobs
with their schedules, e.g., "generate daily reports at 2AM, clean up expired sessions every hour, send
weekly digest emails every Monday 9AM"].

Build a robust scheduled job system that:

Uses [node-cron / APScheduler / Quartz / Sidekiq Cron] to define all jobs as code (not DB config)
Ensures each job runs exactly once in a distributed environment using [Redis lock / DB-level locking]
Implements job observability: logs start time, end time, duration, and outcome for every run
Handles job failure: catches errors, logs them, and alerts on repeated failures
Supports manual trigger: an admin endpoint to fire any scheduled job on demand
Implements graceful shutdown: running jobs complete before the process exits
Provides a job history UI or log query to audit past runs
Show the job definitions, the distributed lock implementation, and the failure alerting integration.
```

47

FEATURE DEVELOPMENT

Implement a bulk data import feature

```
You are a full-stack engineer. I need to let users bulk import [describe data: users,
products, transactions] from a CSV or Excel file.

Build a complete bulk import system that:

Accepts file upload via API: validate file type and size before processing
Parses CSV/XLSX and maps columns to internal schema — supports flexible column ordering
Validates every row: collect all errors (not just the first), report row number and field name per
error
Processes valid rows in batches of 500 using bulk insert (not row-by-row) for performance
Reports progress in real time via WebSocket or SSE: rows processed / total, errors so far
Generates a results report downloadable as CSV: success rows, error rows with error descriptions
Handles large files (>100k rows) without timing out: processes asynchronously as a background job
Show the upload endpoint, the parser and validator, the batch insert logic, and the error report
generator.
```

48

DOCUMENTATION

Write a technical architecture decision record (ADR)

```
Act as a staff engineer. I need to document an architectural decision I made:
[describe the decision, e.g., "switching from a monolith to microservices" / "choosing
PostgreSQL over MongoDB" / "adopting event-driven architecture"].

Write a formal Architecture Decision Record (ADR) that includes:

Title: ADR-[number]: [short imperative title]
Status: [Proposed / Accepted / Deprecated / Superseded by ADR-X]
Context: the forces at play — the problem, constraints, and why a decision was needed
Decision: what was decided in a single clear statement
Alternatives considered: at least 2 other options with pros and cons of each
Consequences: both positive outcomes and accepted tradeoffs / risks
Implementation notes: key steps needed to execute this decision
Write in a neutral, factual tone. This document should be readable by engineers who join the team in 2
years
and need to understand why this choice was made.
```

49

DEVELOPER EXPERIENCE

Generate environment configuration management

```
You are a backend engineer. My application runs across [dev / staging / production]
environments and I need a clean, secure, and maintainable approach to environment-specific
configuration.

Build a configuration management system that:

Defines all config values in a typed config schema with descriptions and defaults
Loads values from environment variables — no config files committed with secrets
Validates all required values at startup — fails fast with clear error messages if any are missing
Provides typed access to config values throughout the app (no raw process.env.X calls in business
logic)
Supports local development via .env file with .env.example as documentation
Handles different value types: strings, numbers, booleans, URLs, durations, arrays
Documents every config variable: what it does, required/optional, and example value
Show the config schema, the loader, the validator, and an example of using the config in a service.
```

50

DATA ENGINEERING

Build a data pipeline

```
Act as a data engineer. I need to build a pipeline that [describe what data flows
where: "extracts user activity from PostgreSQL, transforms it, and loads it into a data warehouse for
analytics"].

Design and implement a complete ETL/ELT pipeline that:

Extracts data from [source: PostgreSQL / REST API / S3 / Kafka] using incremental extraction
(watermark-based, not full-table)
Transforms the data: cleaning, type casting, deduplication, business rule application — each transform
as a
pure function
Loads data to [destination: BigQuery / Snowflake / Redshift / ClickHouse] using bulk load for
efficiency
Schedules the pipeline to run [frequency] and handles late-arriving data correctly
Implements idempotent runs: re-running the same pipeline window produces the same output
Adds data quality checks: null rate, row count reconciliation, schema drift detection
Alerts on failures and data quality failures via [Slack / PagerDuty]
Use [dbt / Apache Airflow / Prefect / Dagster / Luigi]. Show the pipeline definition, transforms, and
quality checks.
```

51

PERFORMANCE

Implement caching with cache invalidation

```
You are a senior engineer. I need to cache [describe what: user profiles, product
catalog, computed recommendations] in my application, but I'm struggling with cache invalidation.

Build a robust caching layer with proper invalidation that:

Identifies the exact staleness tolerance for each data type — TTL strategy derived from business
requirements
Implements tag-based invalidation: group cache keys by entity type so invalidating an entity clears
all related keys
Uses Redis with the right data structures for the job
Implements cache warming for critical paths
Handles thundering herd on cache miss
Adds cache metrics: hit rate, miss rate, eviction count
Documents cache key naming convention: {prefix}:{entity}:{id}:{variant}
Show the cache service, invalidation logic, and repository integration.
```

52

ARCHITECTURE

Design an API gateway

```
Act as an infrastructure architect. I need to set up an API gateway in front of my microservices to handle cross-cutting concerns.

Design and configure an API gateway that:

Routes requests to the correct upstream service based on path prefix or subdomain
Handles authentication and authorization at the edge
Implements rate limiting per client with configurable limits per route
Transforms requests and responses, adding correlation IDs and normalizing errors
Handles SSL termination and keep-alive configuration
Implements circuit breaker for unhealthy upstreams
Produces structured access logs and exports metrics to Prometheus
Show the full gateway configuration, routing rules, and plugin stack.
```

53

TESTING

Write integration tests

```
You are a QA engineer. I need integration tests for my API/service that test the full stack against real dependencies.

Write a complete integration test suite that:

Spins up real dependencies using Testcontainers or Docker Compose before tests run
Tests full request-to-database flows
Covers critical user journeys end to end
Validates HTTP status, response shape, and database state
Tests failure scenarios using mocks/stubs
Cleans up test data between tests
Runs in CI in under 2 minutes with parallelization
Show the test setup, complete scenarios, and CI configuration.
```

54

FEATURE DEVELOPMENT

Build an admin dashboard backend

```
Act as a full-stack engineer. I need to build a backend for an admin dashboard giving internal staff visibility and control.

Build a secure admin backend that:

Requires a separate admin authentication flow with MFA
Implements admin-specific RBAC with permissions per resource
Provides immutable audit logging for every admin action
Builds search, filtering, and CSV export endpoints
Implements bulk actions with confirmation steps
Adds admin impersonation with audit trail
Rate limits and IP-restricts all admin endpoints
Show the admin auth flow, permission middleware, audit schema, and a CRUD module.
```

55

RESILIENCE

Implement a circuit breaker

```
You are a resilience engineering expert. My service calls external dependencies that sometimes go down, causing cascading failures.

Implement a circuit breaker pattern that:

Tracks success/failure counts in a sliding window
Opens the circuit when error rate exceeds a threshold, returning fallback responses
Implements half-open state to test recovery
Closes on successful probe, re-opens on failure
Provides a fallback response per protected call
Exposes circuit state metrics
Logs all state transitions
Show the circuit breaker wrapper applied to an external API call.
```

56

COMPLIANCE & SECURITY

Design a GDPR-compliant data management system

```
Act as a compliance engineer. My application stores EU user data and needs to be GDPR-compliant.

Build a GDPR compliance system that implements:

Right to Access: export all data about a user
Right to Erasure: delete or anonymize all user data
Right to Rectification: update inaccurate personal data
Consent management with timestamp and purpose tracking
Automated data retention policies
Data subject request tracking
Privacy-by-design PII field identification
Show the erasure implementation, consent schema, and retention job.
```

57

DEVELOPER EXPERIENCE

Build a developer portal

```
You are a developer experience engineer. I want to build a developer portal so external developers can discover and integrate with my APIs.

Design a developer portal that includes:

Searchable API catalog
Interactive API reference with try-it-out functionality
Self-service API key management
Usage dashboard per developer
Getting started guides with code samples
Auto-generated SDK downloads
Versioned changelog and migration guides
Outline the tech stack and key pages to build first.
```

58

CONCURRENCY

Implement a distributed lock

```
Act as a backend engineer. I have a distributed system where I need to ensure only one instance processes a critical section at a time.

Implement a distributed locking mechanism that:

Uses Redis or a similar mechanism with a unique owner token
Sets an appropriate lock TTL
Implements lock extension for long-running operations
Releases the lock atomically using a Lua script
Handles lock holder crashes safely
```

```
Wraps the locking logic in a reusable decorator
Show the lock implementation and usage pattern.
```

59

DEBUGGING

Write a memory leak debugging guide

```
You are a performance engineer. My application has a memory leak that grows continuously and forces
periodic restarts.

Guide me through diagnosing and fixing the memory leak:

Instrument the application to capture heap snapshots periodically
Analyze snapshots for unboundedly growing objects and their retainer chains
List the most common causes of leaks in the runtime and match against my profile
Show the correct implementation for each cause
Implement the fix with before/after code
Add memory monitoring and alerting
Add a regression test verifying memory is released
```

60

DEVOPS

Implement a health check system

```
Act as a reliability engineer. I need health check endpoints for my microservices that give
orchestrators accurate visibility.

Build a comprehensive health check system that:

Implements /health/live and /health/ready endpoints
Distinguishes liveness from readiness correctly
Verifies all required dependencies are reachable
Returns structured JSON status with per-component detail
Implements a degraded state
Caches results briefly to prevent overload
Specifies correct Kubernetes probe configuration
```

61

DEVELOPER EXPERIENCE

Generate a monorepo setup

```
You are a developer tooling engineer. I want to migrate my separate repos into a monorepo.

Design and set up a monorepo that:

Defines a clear workspace structure (apps/, packages/, tools/)
Shares code between packages via common libraries
Implements incremental builds that only rebuild affected packages
Runs tasks in correct dependency order, in parallel where safe
Configures workspace import aliases
Sets up versioning strategy
Provides CI that runs only affected pipelines per PR
```

62

FEATURE DEVELOPMENT

Build a data validation library

```
Act as a software engineer. I need a robust data validation library for user input, API payloads, and config files.

Build a validation system that:

Defines schemas for all major data types
Validates primitives, nested objects, arrays, optional fields, unions
Provides field-path-based error messages
Collects all validation errors in one pass
Supports custom business-rule validators
Transforms and sanitizes data on parse
Is shared between server and client validation
```

63

PERFORMANCE

Design a content delivery strategy

```
You are a performance engineer. I need to optimize how my application serves static and dynamic content globally.

Design a content delivery strategy that:

Identifies what belongs on a CDN vs. origin
Configures correct Cache-Control headers per asset type
Implements asset fingerprinting for safe long-term caching
Configures edge caching for SSR/ISR/dynamic pages
Sets up smart cache purging on deploy
Implements image optimization at the edge
Tracks Core Web Vitals impact
```

64

API DESIGN

Build an OpenAPI-first API

```
Act as an API architect. I want to adopt an API-first workflow, defining the OpenAPI spec first and generating code from it.

Set up a complete API-first workflow that:

Writes a full OpenAPI 3.1 spec with paths, schemas, and examples
Configures code generation for server stubs, client SDKs, and types
Sets up spec linting to enforce style rules
Implements contract testing against the running server
Generates auto-updating interactive documentation
Defines a spec governance process
Provides a mock server from the spec
```

65

RESILIENCE

Implement retry logic

```
You are a reliability engineer. My application calls external services that sometimes fail transiently.

Implement a comprehensive retry system that:

Distinguishes retryable from non-retryable errors
Implements exponential backoff with jitter
Sets a max retry count and total timeout budget
Logs every retry attempt
Implements a retry budget to protect downstream services
Wraps retryable operations in a reusable decorator
Show the retry implementation applied to a DB call and an HTTP call.
```

66

FEATURE DEVELOPMENT

Build a reporting module

```
Act as a full-stack engineer. I need to build a reporting module generating financial, activity, or operational reports.

Build a complete reporting system that:

Queries pre-aggregated data for fast generation
Implements a flexible report definition schema
Generates reports in PDF, Excel, and CSV
Schedules automated report delivery via email
Implements on-demand generation with progress tracking
Caches generated reports for repeat parameters
Provides a report history UI
```

67

TESTING

Design a testing pyramid

```
You are a QA architect. My team's test suite is slow, flaky, and low-confidence.

Design a complete testing strategy that:

Maps the testing pyramid ratio for my application type
Defines what belongs in each test layer
Audits the current suite for misplaced tests
Sets test quality standards
Configures parallelization for a fast CI runtime
Implements mutation testing to validate suite effectiveness
Defines a coverage policy
```

68

COMPLIANCE & SECURITY

Implement audit logging

```
Act as a security engineer. I need audit logging for compliance and security investigations.

Build an audit logging system that:

Captures all data-modifying operations
Logs who, what, when, and before/after state
Writes to an append-only store
Implements log integrity via hash chaining
Indexes logs for efficient querying
Provides an admin search and export UI
Defines a retention policy
```

69

FEATURE DEVELOPMENT

Build a payment integration

```
You are a backend engineer. I need to integrate a payment provider to accept payments or subscriptions.

Build a production-ready payment integration that:

Implements the correct payment intent/confirmation flow
Never stores raw card data
Implements idempotency keys on all payment calls
Handles all relevant webhook events with signature validation
Implements daily reconciliation against provider records
Handles refunds, partial refunds, and disputes
Show the payment flow, webhook handler, and reconciliation job.
```

70

FEATURE DEVELOPMENT

Implement a search autocomplete

```
Act as a frontend and backend engineer. I need fast, relevant search autocomplete.

Build a complete autocomplete system that:

Implements debouncing and request cancellation on the frontend
Returns suggestions in under 100ms using a dedicated index
Indexes candidates for prefix and fuzzy matching
Ranks results by relevance
Highlights matched portions
Handles full keyboard navigation
Show the frontend component, backend endpoint, and indexing strategy.
```

71

ARCHITECTURE

Design a permission system

```
You are a software architect. I need a flexible permission system beyond simple RBAC.

Design a permission system that:

Evaluates RBAC vs. ABAC vs. ReBAC for my use case
Models permissions as action-on-resource with conditions
Defines role-to-permission bundling
Implements a centralized can(user, action, resource) check
Caches permission decisions per request
Supports permission inheritance
Provides a permission debugging tool
```

72

ENGINEERING PRACTICES

Build an A/B testing framework

```
Act as a product engineer. I want to run A/B tests to make data-driven product decisions.

Build an A/B testing framework that:

Assigns users to variants consistently via hashing
Supports configurable traffic splits
Logs exposure events
Integrates with an analytics platform for conversion tracking
Provides a statistical significance calculator
Implements guardrail metrics with auto-stop
Show the assignment logic and significance test calculation.
```

73

DATABASE

Implement soft deletes

```
You are a backend engineer. I want to implement soft deletes for audit and recovery purposes.

Implement soft delete that:

Adds a deleted_at column to relevant tables
Creates a global query scope filtering deleted records
Implements restore functionality
Handles cascading soft deletes
Implements hard delete for GDPR erasure
Handles unique constraints among non-deleted rows
Creates a scheduled cleanup job
```

74

ARCHITECTURE

Generate a system design interview answer

```
Act as a senior staff engineer. I have a system design interview question to answer.

Give a structured system design answer that covers:

Requirements clarification (functional and non-functional)
Scale estimation
High-level architecture
Database design and choice justification
Key API endpoints
A deep dive on the hardest component
Bottlenecks and mitigations
```

75

DEVOPS

Write a deployment runbook

```
You are a senior engineer. I need a deployment runbook so any engineer can safely deploy to
production.

Create a complete deployment runbook that includes:

Prerequisites checklist
Pre-deployment steps
Exact deployment procedure with expected output
Verification steps
Rollback procedure
Known issues and workarounds
Escalation contacts
```

76

FEATURE DEVELOPMENT

Build a tenant onboarding automation

```
Act as a backend engineer. I need to automate SaaS customer onboarding to be fully set up within
seconds of signup.

Build a tenant onboarding automation that:

Triggers on signup completion
Provisions tenant resources in the correct order
Runs as a background job with status tracking
Sends a welcome email once complete
Handles partial failures with alerts and manual retry
Is idempotent
Tracks onboarding duration and success rate
```

77

REAL-TIME

Implement a GraphQL subscription

```
You are a GraphQL engineer. I need real-time updates via GraphQL subscriptions.

Implement GraphQL subscriptions that:

Define the subscription type with correct payload
Set up the WebSocket server using graphql-ws
Implement pub/sub for multi-server support
Authenticate WebSocket connections
Scope events to the correct user
Handle subscription lifecycle cleanup
Are tested with a WebSocket client
```

78

DEVOPS

Design a disaster recovery plan

```
Act as an SRE engineer. I need a disaster recovery plan meeting specific RPO/RTO requirements.

Create a disaster recovery plan that:

Identifies all failure scenarios
Maps each scenario to recovery steps and time
Defines the backup strategy
Implements automated database failover
Provides runbooks per failure scenario
Defines a DR testing schedule
Establishes a communication plan
```

79

DEVELOPER TOOLING

Build a changelog generator

```
You are a developer experience engineer. I want to automate changelog generation from git commits.

Build an automated changelog system that:

Enforces Conventional Commits via a git hook
Generates a grouped CHANGELOG.md since the last tag
Auto-increments semantic version correctly
Creates a release with the changelog as the body
Includes migration notes for breaking changes
Generates a version bump commit and tag in CI
Publishes a summary to a team channel
```

80

DEVELOPER TOOLING

Write a code generator

```
Act as a developer tooling engineer. I need a code generator that scaffolds new resources
consistently.

Build a code generator that:

Prompts for inputs with validation
Generates all required files from templates
Derives correct naming conventions from the input name
Auto-registers generated routes/exports
Runs formatters on generated files
Prints a summary with next steps
Is runnable as a single command
```

81

ARCHITECTURE

Implement dependency injection

```
You are a software architect. My codebase has tightly coupled classes that are hard to test.

Refactor the codebase to use dependency injection:

Identify all dependencies that should be injected
Convert classes to constructor injection
Define interfaces for each dependency
Set up a DI container to wire the dependency graph
Show how to swap real implementations for mocks in tests
Handle any circular dependencies
Demonstrate a before/after comparison
```

82

DEVOPS

Design a blue-green deployment strategy

```
Act as a DevOps engineer. I want blue-green deployments for zero-downtime releases.

Design a blue-green deployment system that:

Maintains two identical environments
Deploys and smoke-tests the idle environment first
Shifts traffic progressively with automated checks at each stage
Defines rollback trigger criteria
Handles backward-compatible database migrations
Automates the full flow in CI/CD
```

83

ARCHITECTURE

Build a command query bus

```
You are a software architect. I want to decouple controllers from business logic using a command/query bus.

Implement a command query bus that:

Defines Command and Query interfaces
Implements a CommandBus and QueryBus
Registers handlers automatically
Implements a composable middleware pipeline
Keeps handlers simple with a single handle() method
Shows the full request-to-response flow
```

84

DEVOPS

Generate a Kubernetes deployment configuration

```
Act as a Kubernetes engineer. I need to deploy my application on Kubernetes in a production-ready way.

Generate complete Kubernetes manifests including:

Deployment with rolling update strategy, probes, and resource limits
Service (ClusterIP/LoadBalancer/Ingress)
ConfigMap and Secret
HorizontalPodAutoscaler
PodDisruptionBudget
NetworkPolicy
ServiceAccount with least-privilege RBAC
```

85

DEVELOPER EXPERIENCE

Implement API mocking for development

```
You are a developer experience engineer. My frontend team is blocked waiting for the backend.

Set up an API mock system that:

Generates mock responses from the OpenAPI spec
Returns realistic seeded data
Simulates response latency and error scenarios
Persists state within a session
Runs as a standalone server and as an in-browser service worker
Auto-updates when the spec changes
```

86

PERFORMANCE

Build a GraphQL data loader

```
Act as a GraphQL performance engineer. My API has N+1 query problems.

Implement DataLoader to batch and cache DB queries:

Create a DataLoader per many-to-one relationship
Implement the batch function with a single WHERE id IN query
Handle missing items correctly
Scope each DataLoader to the request context
Enable request-level caching
Add batch-size and cache-hit instrumentation
```

87

DEVOPS

Design a configuration drift detection system

```
You are a platform engineer. My production config has drifted from version control.

Build a configuration drift detection system that:

Defines desired state in version control
Continuously polls actual state and compares
Detects and reports drift with expected vs. actual
Runs checks on a schedule and on every deployment
Alerts on unexpected drift
Distinguishes expected from unexpected drift
Provides one-command remediation
```

88

API DESIGN

Implement a bulk API endpoint

```
Act as a backend engineer. My clients need to batch many individual API calls into one request.

Build a bulk API endpoint that:

Accepts an array of operations
Processes independent operations in parallel
Returns a per-operation result array
Validates each sub-request independently
Limits batch size with a clear error
Returns 207 Multi-Status for partial failure
Optionally supports fully transactional bulk mode
```

89

ENGINEERING PRACTICES

Write a post-mortem report

```
Act as a senior engineer. We had a production incident that needs documenting.

Write a formal post-mortem report that includes:

Incident summary with severity and impact
Timestamped timeline
Root cause analysis (technical and systemic)
Five Whys analysis
What went well
What went wrong
Specific, owned, time-bound action items
```

90

DEVOPS

Implement service mesh configuration

```
You are a platform engineer. I want to add a service mesh for observability, security, and traffic management.

Configure a service mesh that:

Installs the control plane and sidecar injection
Implements mutual TLS between services
Configures circuit breaker, retry, and load-balancing policies
Sets up canary deployments via traffic weights
Configures distributed tracing and metrics export
Implements service-to-service authorization policies
```

91

DEVELOPER EXPERIENCE

Build a developer feedback collection system

```
Act as an engineering productivity engineer. I want structured feedback on internal tools and DX.

Build a developer feedback system that:

Runs a weekly automated survey
Asks targeted follow-up questions on low scores
Aggregates responses and calculates trends
Identifies top pain points by frequency
Publishes a monthly DX report
Auto-creates tickets for top issues
Measures the impact of improvements
```

92

ARCHITECTURE

Implement a tiered storage system

```
You are a storage architect. My storage costs are growing as data ages.

Design a tiered storage system that:

Defines hot/warm/cold tiers and age thresholds
Implements a scheduled tier-transition job
Moves data without loss and updates metadata pointers
Implements transparent retrieval across tiers
Handles cold-tier retrieval latency gracefully
Measures cost savings per tier
Implements lifecycle auto-deletion policies
```

93

ARCHITECTURE

Build a cross-service event schema registry

```
Act as a platform engineer. My microservices' events drift out of sync without a shared schema registry.

Build a schema registry system that:

Defines versioned event schemas centrally
Enforces schema evolution compatibility rules
Registers schemas and assigns schema IDs
Has producers validate against the registry before publishing
Has consumers resolve schemas by ID
Generates type-safe code from schemas
```

94

RELIABILITY

Implement graceful shutdown

```
You are a backend engineer. My application drops in-flight requests on SIGTERM.

Implement graceful shutdown that:

Listens for SIGTERM/SIGINT to start the shutdown sequence
Stops accepting new connections immediately
Waits for in-flight requests up to a timeout
Stops the job queue worker cleanly
Closes DB, cache, and MQ connections
Logs a shutdown summary
```

95

DEVOPS

Design a cost optimization strategy

```
Act as a FinOps engineer. My cloud bill is growing and needs optimization.

Deliver a cloud cost optimization plan that:

Audits the bill by top cost drivers
Identifies idle and oversized resources
Recommends rightsizing based on utilization
Designs a Reserved/Committed Use strategy
Implements autoscaling to reduce over-provisioning
Configures storage lifecycle policies
Sets up budget alerts
```

96

ENGINEERING PRACTICES

Build a feature usage analytics system

```
You are a product engineer. I want to track feature usage to guide product decisions.

Build a feature usage analytics system that:

Defines a simple event tracking API
Uses a consistent event naming convention
Batches events asynchronously
Stores events in an analytics-optimized store
Builds a feature adoption dashboard
Implements funnel analysis
Alerts on features falling below a usage threshold
```

97

SECURITY

Implement cross-origin resource sharing (CORS)

```
Act as a security engineer. I need to configure CORS correctly on my API.

Implement a secure CORS configuration that:

Defines an exact allowlist of origins (never wildcard with credentials)
Configures allowed methods and headers per route
Handles preflight OPTIONS requests correctly
Sets Access-Control-Allow-Credentials only where needed
Sets preflight cache max-age
Validates origin dynamically per environment
Returns clear errors on rejection
```

98

RELIABILITY

Build a dead letter queue handler

```
You are a messaging systems engineer. My DLQ has failed messages needing investigation and
reprocessing.

Build a DLQ management system that:

Reads and enriches DLQ messages with failure context
Categorizes failures as transient, permanent, or unknown
Implements a retry strategy per category
Provides an admin API for viewing and retrying messages
Sends a daily DLQ digest
Auto-purges old messages after archiving
```

99

ARCHITECTURE

Implement a custom ORM layer

```
Act as a backend engineer. I want a lightweight query builder using a bare database driver.

Build a thin query builder that:

Provides a fluent chainable interface
Generates fully parameterized SQL
Supports SELECT, INSERT, UPDATE, DELETE, and JOINs
Maps result rows to typed model objects
Handles transactions with automatic rollback
Logs generated SQL in development mode only
```

100

DATABASE

Build a zero-downtime database migration tool

```
You are a database engineer. I need large-scale schema changes on a live database with zero downtime.

Build a zero-downtime migration tool that:

Determines the minimum safe steps using expand-contract
Adds new columns/tables nullable first
Backfills data in controlled batches
Uses CREATE INDEX CONCURRENTLY for new indexes
Adds constraints as NOT VALID then validates separately
Supports a dual-write cutover period
Drops old schema only after traffic verification
```

101

FEATURE DEVELOPMENT

Implement a content moderation system

```
Act as a backend engineer. My platform has user-generated content needing moderation.

Build a content moderation system that:

Implements pre- and post-moderation based on risk
Integrates with an AI classification service
Defines confidence thresholds for auto-approve/reject/review
Implements an appeal flow
Builds a moderation queue UI
Tracks decisions to retrain the model
Measures false positive/negative rates
```

102

ARCHITECTURE

Design a multi-region active-active architecture

```
You are a distributed systems architect. My application needs to run active-active across multiple regions.

Design a multi-region active-active architecture that:

Routes users to the nearest region
Classifies and replicates data appropriately per type
Implements cross-region database replication
Handles write conflicts with an appropriate resolution policy
Implements health-based automatic failover
Manages session continuity across regions
Calculates cost vs. resilience tradeoffs
```

103

FEATURE DEVELOPMENT

Build a dynamic form builder backend

```
Act as a full-stack engineer. I need a dynamic form builder backend like Typeform, embedded in my platform.

Build a form builder backend that:

Defines a flexible form schema model
Supports common field types
Implements conditional field visibility logic
Validates submissions against the schema
Stores submissions linked to schema version
Generates a submission report with CSV export
```

104

ARCHITECTURE

Implement a serverless function architecture

```
You are a cloud architect. I want to migrate a workload to serverless functions.

Design a serverless architecture that:

Identifies which workloads fit serverless
Decomposes into single-responsibility functions
Handles cold starts appropriately
Manages shared state externally
Configures function-to-function communication
Sets memory, timeout, and concurrency limits
Implements observability with cost tracking
```

105

PERFORMANCE

Generate a performance budget

```
Act as a performance engineer. My web app has degraded over time and needs enforced performance budgets.

Create a performance budget system that:

Measures the current Core Web Vitals baseline
Defines budget thresholds per metric
Integrates budget checks into CI
Generates a performance report on every merge
Identifies and prioritizes the biggest violations
Implements real-user monitoring
Creates a performance regression playbook
```

106

FEATURE DEVELOPMENT

Build a SaaS billing system integration

```
You are a backend engineer. I need to integrate subscription billing into my SaaS application.

Build a complete billing integration that:

Implements the full subscription lifecycle
Handles all relevant billing webhooks
Implements metered usage billing
Handles failed payments with dunning and suspension
Implements a trial period with reminders
Provides a customer billing portal
Implements correct proration on upgrades
```

107

BACKEND DEVELOPMENT

Implement a message queue consumer group

```
Act as a messaging systems engineer. I need a consumer group processing messages at scale with
multiple workers.

Build a consumer group implementation that:

Configures partition/shard assignment across workers
Commits offsets only after successful processing
Handles rebalancing gracefully
Processes messages in batches with per-message error isolation
Implements backpressure
Sends unprocessable messages to a DLQ
Exposes consumer lag metrics
```

108

DATABASE

Design a read replica routing strategy

```
You are a database engineer. All queries currently hit the primary despite having read replicas.

Implement a read/write splitting strategy that:

Routes all writes to the primary automatically
Routes reads to replicas by default
Handles replication lag with a consistency override
Implements replica health checks
Load balances reads across healthy replicas
Wraps the routing transparently in the DAL
Monitors and alerts on replication lag
```

109

SECURITY

Build a request validation and sanitization layer

```
Act as a security-focused backend engineer. I need centralized request validation to prevent injection
attacks.

Build a validation and sanitization system that:

Validates all requests against a schema before controller logic
Sanitizes string inputs against XSS
Rejects unexpected extra fields
Validates and sanitizes file uploads
Escapes values used in SQL, shell, or regex construction
Fails fast before any DB or external calls
Logs rejected requests for abuse detection
```

110

ARCHITECTURE

Implement a saga pattern for distributed transactions

```
You are a distributed systems architect. A business process spans multiple microservices and needs transactional consistency.

Implement the saga pattern that:

Evaluates choreography vs. orchestration for my use case
Defines each step with a compensating action
Implements the saga orchestrator with state tracking
Handles partial failure with reverse-order compensation
Persists saga state to survive restarts
Implements idempotent step handlers
Provides saga execution observability
```

111

PERFORMANCE

Write a database connection pooling configuration

```
Act as a backend performance engineer. My application is hitting connection exhaustion errors under load.

Diagnose and configure database connection pooling that:

Calculates the correct pool size for my workload
Sets appropriate min/max pool size
Configures connection, idle, and lifetime timeouts
Implements connection validation (test-on-borrow)
Ensures connections are always released
Separates read and write pools if using replicas
Adds pool metrics
```

112

FRONTEND DEVELOPMENT

Build a client-side state management architecture

```
You are a frontend architect. My application's state management has become unpredictable.

Design a state management architecture that:

Separates server state, client state, and URL state
Implements server state with caching and optimistic updates
Implements client state with a normalized shape
Defines clear read/mutation boundaries
Prevents unnecessary re-renders
Handles derived state with computed selectors
Documents the architecture with a diagram
```

113

SECURITY

Implement API request signing

```
Act as a security engineer. I need to secure server-to-server API calls using request signing.

Implement HMAC-based request signing that:

Generates a signature from method, path, timestamp, and body hash
Includes signature, timestamp, and key ID in headers
Validates using constant-time comparison
Rejects stale timestamps to prevent replay
Supports key rotation
Provides a client-side signing helper
Logs repeated verification failures
```

114

FEATURE DEVELOPMENT

Design a multi-step checkout flow backend

```
You are a backend engineer for an e-commerce platform. I need a multi-step checkout flow backend.

Build a checkout backend that:

Persists checkout state server-side with a TTL
Validates cart contents at every step
Calculates totals server-side, never trusting client values
Reserves inventory temporarily during checkout
Creates the order atomically with payment confirmation using idempotency keys
Triggers downstream fulfillment only after payment succeeds
Handles abandoned checkout recovery
```

115

TESTING

Build a code coverage enforcement pipeline

```
Act as a QA engineering lead. I want meaningful coverage standards without encouraging low-value
tests.

Set up a coverage enforcement system that:

Configures branch coverage collection
Sets differentiated thresholds for new vs. legacy code
Fails PRs only on newly added uncovered code
Excludes generated code and trivial code appropriately
Posts a coverage diff comment on PRs
Tracks coverage trend over time
Documents guidance against gaming the metric
```

116

ARCHITECTURE

Implement an outbox pattern for reliable event publishing

```
You are a distributed systems engineer. My service can lose events if it crashes between the DB write
and publish.

Implement the transactional outbox pattern that:

Writes the data change and event to an outbox table in one transaction
Implements a relay process that polls and publishes unpublished events
Marks events published only after broker acknowledgment
Considers CDC as a lower-latency alternative to polling
Ensures idempotent consumers for at-least-once delivery
Cleans up published events on a schedule
Monitors outbox growth and publish lag
```

117

CODE QUALITY

Write a code complexity analysis report

```
Act as a code quality engineer. I want to analyze the codebase for complexity hotspots.

Analyze the codebase and generate a report that:

Calculates cyclomatic complexity per function and flags high values
Identifies oversized files and functions
Detects near-duplicate code blocks
Measures module coupling
Cross-references complexity with change frequency
Prioritizes a refactoring backlog
Sets ongoing complexity budgets in CI
```

118

ARCHITECTURE

Build a request context propagation system

```
Act as a backend engineer. I need request-scoped context propagated without threading it through every function.

Implement request context propagation that:

Uses the runtime's context mechanism to store request-scoped values
Sets context at the entry-point middleware
Makes context accessible anywhere in the call stack
Propagates correctly across async boundaries
Auto-injects context into every log statement
Propagates trace context to outgoing calls
Verifies no context leakage under concurrent load
```

119

API DESIGN

Design an API deprecation and sunset process

```
You are an API product engineer. I need to formally deprecate and remove an old endpoint or version.

Design a deprecation process that:

Announces deprecation with adequate notice
Adds Deprecation and Sunset headers to responses
Logs calls to build a migration outreach list
Sends direct notifications to affected users
Provides a clear migration guide
Tracks migration progress via a usage dashboard
Returns HTTP 410 Gone on the sunset date
```

120

DATABASE

Implement a rollback-safe blue/green database schema strategy

```
Act as a database engineer. I need a schema change that's safe to roll back even after the app deploys.

Design a rollback-safe schema migration that:

Adds new columns/tables without removing old ones
Ensures backward compatibility with the previous app version
Uses dual writes during the transition
Provides a verified rollback script
Defines a "safe to clean up" checklist
Runs cleanup as a separate later deployment
Documents the full migration timeline
```

Want more prompts like these?

Explore Folio3's full library of curated AI prompts across software development, marketing, healthcare, and more — built to help teams get better output from every model, faster.

Browse the Prompt Directory

https://www.folio3.ai/prompts/chatgpt/software-development