SpecRabbit

Visual Application Architecture Tool

User Guide

Contents

1. Purpose and Overview

SpecRabbit is a visual application architecture specification tool. The user constructs an application model as a graph of typed nodes connected by flows. The resulting specification is exported as a self-describing JSON or YAML document that an LLM can consume to generate production-ready code.

The tool operates in two distinct areas:

- Graph canvas – nodes representing application layers (UI forms, API endpoints, backend components, databases, etc) and flows connecting them.
- Project-wide parameters – settings that apply globally to the whole project, including technology stack choices (framework, CI/CD, etc.).

AI code generation tools will use provided application graph and project-wide parameters as guidelines for code generation.

2. Data Type System

A formal data type system is shared across API endpoint parameters, backend component parameters and return values, database query parameters, and database table fields.

2.1 Primitive Data Types

Attribute	Description
string	UTF-8 text of arbitrary length.
integer	Signed whole number.
float	Signed floating-point number.
boolean	True or false.
datetime	Date and time.
uuid	Unique global identifier
blob	Binary data (file, image, raw bytes).
enum	Fixed list of values.
any	Untyped – any data type can be here.
custom	User-defined primitive; requires a non-empty custom type name string that does not coincide with any reserved primitive name.

2.2 Structure Data Type

A structure is an ordered set of named fields. Each field has name*, data type* that is either a primitive, shared data type (see below), or another structure, enabling arbitrary nesting depth. A field also has optional description. Structures have no standalone name – they are defined inline wherever a data type is required. Structures and shared types should have at least one field.

2.3 Array Data Type

Primitive data type or structure may represent array of items with that data type, rather than a single value. Therefore, data type has a boolean attribute indicating if this data type should be treated as array:

Attribute	Description
is array	Boolean flag indicating if this data type represents an array.

2.4 Shared Data Type

Shared type is a reusable structure that can be referred from different places of the project:

Attribute	Description
name*	Unique identifier for this shared data type node within the project.
description	Free-text description and any additional information about the shared data type.
fields*	Data type structure as set of fields. Each field has a name*, data type*, and description. At least one field must be provided.

2.5 Data Type Categories

Each data type used in the project can be one of the following possible categories:

- **Primitive** – specific primitive data type should be selected from predefined list. If custom is selected, then the name of the custom data type must be provided.
- **Structure** – structure of the data type is defined in-line
- **Shared** – reference of actual shared data type is provided.

3. Node Types

The canvas supports seven node types. Each node and subnode (such as control subnode of UI form node) is identified by a unique mandatory name attribute. Each node and subnode also has a description attribute – all structured and non-structured details about the node or subnode must be captured there. Node types may have additional attributes on top of name and description. Mandatory attributes are indicated by star (“*”) character after attribute name.

3.1 UI Form

UI form node represents a user interface screen or dialog that the user interacts with. If project has at least one UI form, the project must have a startup form assigned, meaning the application must start with that UI form.

Form-level attributes

Attribute	Description
name*	Unique identifier for this form within the project.
title	Display title shown to the end user on the rendered form.
description	Free-text description and any additional information about the form.
type*	"Regular" (full-page form) or "Modal" (popup dialog).
validation	Free-text description of client-side cross-field data validation performed before sending request to backend. For single field validation, use control-level validation attribute (see below).

Controls

A form contains a set of controls. More controls can be added by AI on top of specified ones during code generation depending on context. Controls may be grouped into sections – only one level of control grouping into sections is allowed.

Section attributes:

Attribute	Description
name*	Unique identifier for this section within the form.
label	Display label for the section (e.g. rendered as a group header).
description	Free-text description and any additional information about the section.

Control attributes:

Attribute	Description
name*	Unique identifier for this control within the UI form. Used when referencing controls during flow definitions.
label	Display label shown next to the control.
type*	Control type – see control type list below.
required	Boolean. When true, the form cannot be submitted unless this control has a value. This attribute is only applicable to data-holding controls (string edit, date edit, number edit, masked edit, checkbox, radio group, dropdown, multi-select, file upload, custom) and is ignored for non-data controls.
readonly	Boolean. When true, the field cannot be changed. Required is ignored if this attribute is set to true. This attribute is only applicable to data-holding controls (string edit, date edit, number edit, masked edit, checkbox, radio group, dropdown, multi-select, file upload, custom) and is ignored for non-data controls.
validation	Free-text description of validation rules to apply to this control's value. This attribute is only applicable to data-holding controls (string edit, date edit, number edit, masked edit, checkbox, radio group, dropdown, multi-select, file upload, custom) and is ignored for non-data controls. For cross-field validation use form-level validation attribute - see above.
description	Free-text description and any additional information about the control.

Control types:

Attribute	Description
string edit	Single-line text input.
masked edit	Single-line masked text input for sensitive data such as passwords.
date edit	Date picker or date-formatted text input.
number edit	Numeric input (integer or decimal).
checkbox	Boolean toggle (checked / unchecked).
radio group	Single selection from a set of mutually exclusive defined options.
dropdown	Single selection from a dropdown list defined for the control.
multi-select	Multiple selections from a list.
file upload	Control allowing the user to upload one or more files (blob).
link	Clickable hyperlink label.
button	Clickable button that triggers a flow.
label	Read-only display text.
image	Read-only image display.

Attribute	Description
custom	User-defined control type. Requires a non-empty custom type name that does not match any reserved control type name.

3.2 API Endpoint

API endpoint node represents an HTTP endpoint exposed by the backend.

Endpoint-level attributes

Attribute	Description
name*	Unique identifier for this endpoint within the project.
description	Free-text description and any additional information about the endpoint.
url*	Full URL path of the endpoint including GET parameters (e.g. /api/v1/users/{id}). It must be unique within a project.
method*	HTTP method: GET, QUERY, POST, PUT, PATCH, or DELETE.
type*	Enum: “Public” – externally available without a need in authentication credentials. “Protected” – externally accessible endpoint requiring valid authentication credentials. “Internal” – not externally available internal endpoint.
return fields	Optional list of returned fields. Each return field has a name*, data type*, and description. More return fields may be added by AI during code generation depending on context.

Parameters

API endpoint can have zero or more parameters. HTTP method determines how parameters are passed – for example, URL is used to pass parameters for GET request. More parameters may be added by AI on top of specified ones during code generation depending on context. Each parameter has:

Attribute	Description
name*	Unique identifier for this parameter within the endpoint.
description	Free-text description and any additional information about the parameter.
data type*	A primitive data type, structure or shared type.
default value	A literal default value for this parameter (free text interpreted by the AI according to the data type).

3.3 Backend Component

Backend component node represents an internal backend service. It is not directly exposed to the outside world, but it can be accessed via flow coming from API endpoint, other backend component, or scheduler. Backend component may have a signature: list of parameters and return value. Inbound flow calls code inside of backend component, passing parameter arguments and handling returned value.

Component-level attributes

Attribute	Description
name*	Unique identifier for this component within the project.
description	Free-text description and any additional information about the component, including algorithm that the component should implement.
return data type	The data type (primitive, structure or shared) returned by this component, or "void" if nothing is returned. Void return data type can still return a status code through the flow bounce-back mechanism.

Parameters

Backend component can have an ordered list of input parameters. More parameters may be added by AI during code generation depending on a context. Each parameter has:

Attribute	Description
name*	Unique identifier for this parameter within the component.
description	Free-text description and any additional information about the parameter.
data type*	A primitive data type, structure or shared type.
default value	A literal default value for this parameter (free text, interpreted by the AI according to the data type).

3.4 Database

Database node represents a SQL or NoSQL database. Database nodes are always terminal nodes in a flow – after a database query executes, the result bounces back to the flow origin.

Database-level attributes

Attribute	Description
name*	Unique identifier for this database node within the project.
description	Free-text description and any additional information about the database.

Attribute	Description
provider*	Specific database provider.

Tables

Database contains a set of tables. Each table has:

Attribute	Description
name*	Unique identifier for the table within database.
description	Free-text description and any additional information about the table.
fields	Ordered list of fields. Each field has: name*, data type*, description (free text) and is_nullable* (boolean). More fields may be added by AI during code generation depending on a context.

Queries

A database node contains a set of queries. When a flow enters a database node, the specific query to execute must be selected for the flow. Each query has:

Attribute	Description
name*	Unique identifier for this query within the database node.
description	Free-text description of what this query does.
type*	Query operation type: Insert, Update, Upsert, Retrieve, Delete, Control, Composite (any combination of other operation types).
parameters	Ordered list of input parameters. Each parameter has: name*, data type*, default value, description (free-text). More parameters may be added by AI during code generation depending on context.
return fields	For retrieve queries only: optional list of fields returned. Each return field has a name*, data type*, and description. More return fields may be added by AI during code generation depending on context.

Relations

A database node may define relations between its tables. Each relation has:

Attribute	Description
name*	Unique identifier for this relation within the database node.
type*	Cardinality: “One-to-one”, “One-to-many”, or “Many-to-many”.
source table*	Reference to existing source table.

Attribute	Description
source field	Field name in the source table, or table's primary key is assumed if value is missing.
destination table*	Reference to existing destination table.
destination field	Field name in the destination table, or table's primary key is assumed if value is missing.
description	Free-text description and any additional information about the relation.

3.5 File Storage

File storage node represents a container of file, such is AWS S3. File storage nodes are always terminal nodes in a flow – after operation in file storage completes, the result bounces back to the flow origin.

File storage attributes

Attribute	Description
name*	Unique identifier for this file storage within the project.
description	Free-text description and any additional information about the file storage.
provider*	Specific file storage provider.

3.6 Scheduler

Scheduler node represents a time-based trigger that initiates a flow on a defined schedule.

Scheduler attributes

Attribute	Description
name*	Unique identifier for this scheduler within the project.
description	Free-text description and any additional information about the scheduler, including schedule of firing.

3.7 Custom Node

Custom node represents architectural objects not covered by any predefined node type. Custom node can be placed anywhere in flow chain. When a flow enters a custom node, the AI infers the interaction from the custom node type, descriptions of the custom node and the flow edge. AI might generate only interfaces and abstract classes without substantial implementation if description of a custom node does not provide enough details to generate complete node implementation.

Custom node attributes

Attribute	Description
name*	Unique identifier for this custom node within the project.
type*	Name of custom node type.
description	Free-text description and any additional information about the custom node.
return data type	The data type (primitive, structure or shared) returned by this custom node, or "void" if nothing is returned. Void return data type can still return a status code through the flow bounce-back mechanism.

Parameters

Custom node can have an ordered list of input parameters. More parameters may be added by AI during code generation depending on a context. Each parameter has:

Attribute	Description
name*	Unique identifier for this parameter within the custom node.
description	Free-text description and any additional information about the parameter.
data type*	A primitive data type, structure or shared.
default value	A literal default value for this custom node (free text, interpreted by the AI according to the data type).

4. Flow

Flow represents a single logical signal propagating through a chain of connected nodes. Flow is a first-class object – it is created and named independently of the node graph, then assigned to edges. The default flow semantic is that data propagates from start node to end node via intermediary nodes, and result is propagated back from end node to start node. A different semantic applies when start node and end node represent different UI forms: it represents transition from one to another UI form without going to backend, and the flow must have only one pair connecting start and end UI form nodes.

Flow attributes

Attribute	Description
name*	Unique identifier name for this flow within the project.
description	Free-text description and any additional information about the flow.
color	Display color used to visually distinguish this flow on the canvas. Stored as a hex color string (e.g. #1D9E75). Has no semantic meaning for code generation – used for human readability only.
start node	The node where this flow originates. Must be UI form, API endpoint, scheduler, custom node, or empty (for flow without edges).
end node	The terminal node where the flow ends. Must be UI form, backend component, database, file storage, custom node, or empty (for flow without edges).

4.1 Flow Edges

A flow is a linear chain of edges. Flow edge connects two nodes in the flow. One flow edge has the following attributes:

Flow edge attributes

Attribute	Description
description	Free-text description and any additional information about this flow edge.
source node*	The source node of the flow edge.
target node*	The destination node of the flow edge.

4.2 Allowed Node Pairs

Each pair in the chain must be one of the following:

From node type	To node type
UI form	UI form, API endpoint (allowed endpoint types: “Public” and “Protected”).
API endpoint	API endpoint, backend component, database, file storage, custom node.
Scheduler	API endpoint, backend component, database, file storage, custom node.
Backend component	API endpoint, backend component, database, file storage, custom node.
Custom node	Any node type.

4.3 Flow End: Database

When flow terminates in database node, query of the database node must be specified for incoming flow edge:

Flow edge attributes

Attribute	Description
query	Reference to a query from destination node when destination node is database. Required attribute when destination node is database, and it must be empty otherwise.

4.4 Flow End: File Storage

When flow terminates in file storage node, operation must be specified for incoming flow edge:

Flow edge attributes

Attribute	Description
operation	Reference to an operation type on destination node when destination node is file storage. The field should take one of the following values: Upload, Download, Delete, List, Get Metadata, Generate Signed URL, Copy, Move, Set Metadata, Check Existence. Required attribute when destination node is file storage, and it must be empty otherwise.

4.5 Flow Start: UI Form

When flow starts from UI form, the following parameters must be specified for a source UI form of the flow (both attributes are required when the start node is a UI form):

Flow attributes

Attribute	Description
initiating control	The specific control on the source UI form that triggers this flow. Selected from the list of controls defined on that form. Form itself is considered iniating event if the field is empty (for example, UI form loading event).
initiating event	Free-text description of the event on that form or control that fires the flow (e.g. "button click", "field blur", "field focus", "value change"). Required when start node is a UI form, and it must be empty otherwise.

4.6 Multiple Outgoing Legs of Backend Component

All nodes in a flow chain must appear exactly once. Consequently, every node may have at most one incoming leg and at most one outgoing leg, and cycles are not permitted. Backend components are the sole exception to the outgoing leg constraint: a backend component may have more than one outgoing leg, and destination nodes allow more than one enterance of such outgoing legs. One purpose of multiple outgoing legs is to allow a backend component to query multiple data sources, other backend components and/or API endpoints in sequence and compose their results before the combined result bounces back to the flow origin.

Multiple outgoing legs from the same backend component are treated as an ordered set of flow pairs sharing the same source node but each having a distinct destination node. Each leg carries its own description and an order number specifying its position in the execution sequence. Legs are invoked sequentially in ascending order number. Failure handling behaviour for a leg is described in that leg's description field.

Flow edge has an attribute to specify order of execution in case of multiple outgoing legs:

Flow edge attributes

Attribute	Description
name	Unique identifier name for this flow within the multiple leg configuration of a single backend component – used and mandatory only for the case of multiple outgoing legs of backend component.

4.7 Flow Leg

The same edge can enter the same destination node via different legs in case of multiple leg configuration.

Flow leg attributes

Attribute	Description
execution ordinal	Required ordinal number of flow execution of this leg. Non-editable field assigned automatically in the order of legs.
description	Free-text description and any additional information about this leg.

4.8 Flow Legs To Database Node

When flow leg enters database node, it must have its own query specified, overriding database query specified in parent flow edge of the leg.

Flow leg attributes

Attribute	Description
query	Reference to a query from destination node when destination node is database. Required attribute when destination node is database, and it must be empty otherwise. Overrides query specified in parent flow edge.

4.9 Return Path To a Different UI Form

When a flow originates from a UI form and goes to backend, but its terminal node is not a UI form, the result bounces back to the originating UI form by default. However, it may be desirable to navigate to a different UI form upon completion rather than returning to the origin. To support this, when the source of a flow is a UI form, an optional return UI form attribute may be specified identifying a different form to display after the result bounces back. If not specified, the result returns to the starting UI form identified by start node.

This is distinct from the case where a flow explicitly connects two UI forms as source and destination – in that case the flow represents a direct navigation transition between forms with no backend involvement.

Flow has an attribute to specify different UI form to return to:

Flow attributes

Attribute	Description
return UI form	Optional attribute specifying a different UI form to navigate to after the flow result bounces back. Applicable only when the start node is a UI form.

5. Project-Wide Parameters

Global parameters capture the technology decisions that apply to the entire application. They are set once per project and exported in the "parameters" section of the spec. Every single-select and multi-select field includes an "Other" option; selecting "Other" reveals a mandatory free-text field for a custom value.

Parameters are organized into eight groups. Each group is independent – changing one group does not affect others, except where cross-field validation rules apply (see Section 6).

5.1 Backend

Parameter	Available values
API Style (single select)	REST (JSON:API Spec) · REST (HAL) · REST (Custom) · GraphQL (Relay) · GraphQL (Apollo Federation) · gRPC (Unary) · gRPC (Streaming) · tRPC (TypeScript) · Other
Versioning Strategy (single select)	URL Path (/v1/) · Custom Header (X-API-Version) · Accept Header · Query Parameter · Date-based (Stripe-style) · No Versioning · Other
Pagination Strategy (single select)	Cursor-based · Offset/Page-based · Keyset · Any (case-by-case) · Other
API Features (multi-select)	Cursor Pagination · Offset Pagination · Advanced Filtering · Multi-field Sorting · Sparse Fieldsets · Resource Expansion/Includes · Batch Operations · Bulk Import/Export · Full-text Search · Faceted Search · Aggregations/Analytics · ETags (Optimistic Locking) · Idempotency Keys · HATEOAS Links · Webhooks · WebSocket Subscriptions · Server-Sent Events · File Upload (Multipart) · Streaming Responses · Other
Rate Limiting Strategy (single select)	Token Bucket (allows bursts) · Sliding Window (smooth) · Fixed Window (simple) · Adaptive (auto-adjusting) · Other
API Security (multi-select)	CORS · CSRF Protection · XSS Prevention · SQL Injection Prevention · Input Validation · Request Signing (HMAC) · Mutual TLS · API Gateway · Web Application Firewall · DDoS Protection · Rate Limiting · Bot Detection · Other
Backend Framework (multiple select)	NestJS · Express · Fastify · Koa · Hapi · Django · FastAPI · Flask · Spring Boot · ASP.NET Core · Laravel · Ruby on Rails · Phoenix (Elixir) · Gin (Go) · Fiber (Go) · Echo (Go) · Actix (Rust) · Axum (Rust) · Other
Backend Language (multiple select)	TypeScript · JavaScript · Python · Java · C# · Go · Rust · Ruby · PHP · Elixir · Kotlin · Swift · Scala · Other
Additional requirements (textarea)	(free text, optional)

5.2 Frontend

Parameter	Available values
Frontend Framework (single select)	React · Angular · Vue.js · Next.js · Remix · Nuxt · Gatsby · Svelte · SvelteKit · SolidJS · Astro · Qwik · SolidStart · Other
Rendering Strategy (single select)	Client-Side Rendering (SPA) · Server-Side Rendering (SSR) · Static Site Generation (SSG) · Incremental Static Regeneration (ISR) · Hybrid (mixed per route) · Other
Styling Solution (single select)	Tailwind CSS · CSS Modules · Styled Components · Emotion · Vanilla Extract · SASS/SCSS · UnoCSS · Other
State Management (single select)	Zustand · Jotai · Redux Toolkit · MobX · XState · TanStack Query (server state) · React Context Only · Other
UI Features (multi-select)	Progressive Web App · Offline Support · Push Notifications · Internationalization · Accessibility (WCAG AA) · Dark Mode · Dynamic Theming · Lazy Loading · Code Splitting · SEO Optimization · Analytics · A/B Testing · Feature Flags · Error Boundaries · Other
Additional requirements (textarea)	(free text, optional)

5.3 Database

Parameter	Available values
Database (multi-select)	PostgreSQL · MySQL · MongoDB · DynamoDB · Cassandra · CockroachDB · PlanetScale (MySQL) · Supabase (PostgreSQL) · Neon (PostgreSQL) · Turso · ClickHouse (Analytics) · Neo4j (Graph) · TimescaleDB (Time-series) · SQLite · Other
ORM (multi-select)	Prisma · TypeORM · Drizzle · Sequelize · Mongoose · SQLAlchemy · Django ORM · Hibernate · Entity Framework Core · ActiveRecord (Rails) · Ecto (Elixir) · GORM (Go) · sqlx (Go/Rust) · Eloquent (Laravel) · None (raw queries) · Other
Search Engines (multi-select)	Elasticsearch · OpenSearch · Algolia · Typesense · Meilisearch · Other
Vector Databases (multi-select)	Pinecone · Weaviate · Qdrant · pgvector · Chroma · Other
Caching (multi-select)	Redis · Redis Cluster · Memcached · ElastiCache · Dragonfly · Other
Message Queues (multi-select)	Kafka · RabbitMQ · Amazon SQS · Google Pub/Sub · NATS · Apache Pulsar · Redpanda · Other
File Storage (multi-select)	Amazon S3 · Google Cloud Storage · Azure Blob · Cloudflare R2 · MinIO · Other
Schema Patterns (multi-select)	Soft Deletes · Audit Columns (created/updated) · Multi-tenancy · Temporal/History Tables · URL Slugs · UUIDs (vs auto-increment) ·

Parameter	Available values
	Polymorphic Relations · JSON/JSONB Columns · Full-text Search Indexes · Row-Level Security · Other
Additional requirements (textarea)	(free text, optional)

5.4 Authentication & Security

Parameter	Available values
Primary Auth Method (single select)	Email/Password · OAuth 2.0/Social Only · Magic Link (Passwordless) · Passkeys (WebAuthn) · Enterprise SSO (SAML/OIDC) · Other
Additional Auth Methods (multi-select)	OAuth 2.0/Social Login · Magic Link · Phone OTP · Passkeys · SAML 2.0 · LDAP/Active Directory · API Keys · Other
OAuth Providers (multi-select)	Google · Apple · Microsoft · GitHub · GitLab · Facebook · Twitter/X · LinkedIn · Discord · Slack · Okta · Auth0 · Azure AD · AWS Cognito · Other
MFA Methods (multi-select)	TOTP (Authenticator App) · SMS · Email · Push Notification · Hardware Key (YubiKey) · WebAuthn · Backup Codes · Other
Security Features (multi-select)	Argon2 Password Hashing · Password Policy · Account Lockout · Brute Force Protection · Password Breach Detection · Suspicious Login Detection · Authentication Audit Log · Role-Based Access Control (RBAC) · Attribute-Based Access Control (ABAC) · Multi-tenant Isolation · Data Encryption at Rest · PII Masking · Other
Compliance Requirements (multi-select)	GDPR · CCPA/CPRA · HIPAA · PCI-DSS · SOC 2 Type II · ISO 27001 · FedRAMP · SOX · NIST CSF · HITRUST · Other
Additional requirements (textarea)	(free text, optional)

5.5 Infrastructure

Parameter	Available values
Cloud Provider (single select)	AWS · Google Cloud · Microsoft Azure · Vercel · Cloudflare · Railway · Fly.io · Render · Netlify · DigitalOcean · Hetzner · Other
Additional Services (multi-select)	Cloudflare CDN · Vercel Edge · AWS CloudFront · Auth0 · Stripe · Twilio · SendGrid · Segment · Other
Container Strategy (single select)	Docker Compose (self-hosted) · Kubernetes (EKS/GKE/AKS) · AWS ECS/Fargate · Google Cloud Run · AWS App Runner · Azure Container Apps · Serverless Functions · PaaS (Heroku-style) · HashiCorp Nomad · Other

Parameter	Available values
Infrastructure as Code (single select)	None (Manual/Console) · Terraform · Pulumi · AWS CDK · CloudFormation · Ansible · Other
CI/CD Platform (single select)	GitHub Actions · GitLab CI · CircleCI · Jenkins · Argo CD (GitOps) · Buildkite · Flux CD · Other
Environments (multi-select)	Local Development · Development · Staging · UAT · Production · Disaster Recovery · Other
Additional requirements (textarea)	(free text, optional)

5.6 Observability

Parameter	Available values
Logging Stack (single select)	Grafana Loki · ELK Stack (Elasticsearch, Logstash, Kibana) · Datadog · AWS CloudWatch · Google Cloud Logging · Splunk · Fluent Bit · Other
Metrics (multi-select)	Prometheus · VictoriaMetrics · Datadog Metrics · CloudWatch Metrics · New Relic · Dynatrace · Other
Tracing Stack (single select)	OpenTelemetry + Jaeger · OpenTelemetry + Tempo · Datadog APM · AWS X-Ray · New Relic APM · Honeycomb · Other
Error Tracking (single select)	Sentry · Bugsnag · Rollbar · Datadog RUM · Crashlytics · LogRocket · Use Logging Only · Other
Alerting Channels (multi-select)	Alertmanager · PagerDuty · Opsgenie · Slack · Email · SMS · Webhooks · incident.io · Other
Additional requirements (textarea)	(free text, optional)

5.7 Testing

Parameter	Available values
Unit Test Framework (single select)	Vitest · Jest · pytest · Go Test · JUnit · Other
E2E Test Framework (single select)	Playwright · Cypress · Selenium · Puppeteer · Other
Testing Strategies (multi-select)	Test-Driven Development · Behavior-Driven Development · Contract Testing · Snapshot Testing · Visual Regression Testing · Load Testing · Chaos Engineering · Accessibility Testing · Other
Performance Testing (multi-select)	k6 · Locust · Gatling · Artillery · Lighthouse · Other

Parameter	Available values
Security Scanning (multi-select)	Snyk · SonarQube · Semgrep · CodeQL · OWASP ZAP · Trivy · Dependabot · Other
Coverage Target (number, 0–100)	Default: 80
Additional requirements (textarea)	(free text, optional)

6. Parameter Validation Rules

Validation fires on parameter change and project load. When "Other" is selected for any field, all validation rules involving that field are suspended. If two rules are violated simultaneously, both error messages are shown.

6.1 Backend Framework → Backend Language

Field / value	Condition	Error message
NestJS	Language must include TypeScript or JavaScript	NestJS is a TypeScript/JavaScript framework. Add TypeScript or JavaScript language, or select a different framework.
Express	Language must include TypeScript or JavaScript	Express is a TypeScript/JavaScript framework. Add TypeScript or JavaScript language, or select a different framework.
Fastify	Language must include TypeScript or JavaScript	Fastify is a TypeScript/JavaScript framework. Add TypeScript or JavaScript language, or select a different framework.
Koa	Language must include TypeScript or JavaScript	Koa is a TypeScript/JavaScript framework. Add TypeScript or JavaScript language, or select a different framework.
Hapi	Language must include TypeScript or JavaScript	Hapi is a TypeScript/JavaScript framework. Add TypeScript or JavaScript language, or select a different framework.
Django	Language must include Python	Django is a Python framework. Add Python language, or select a different framework.
FastAPI	Language must include Python	FastAPI is a Python framework. Add Python language, or select a different framework.
Flask	Language must include Python	Flask is a Python framework. Add Python language, or select a different framework.
Spring Boot	Language must include Java or Kotlin	Spring Boot is a JVM framework. Add Java or Kotlin language, or select a different framework.

Field / value	Condition	Error message
ASP.NET Core	Language must include C#	ASP.NET Core is a C# framework. Add C# language, or select a different framework.
Laravel	Language must include PHP	Laravel is a PHP framework. Add PHP language, or select a different framework.
Ruby on Rails	Language must include Ruby	Ruby on Rails is a Ruby framework. Add Ruby language, or select a different framework.
Phoenix (Elixir)	Language must include Elixir	Phoenix is an Elixir framework. Add Elixir language, or select a different framework.
Gin (Go)	Language must include Go	Gin is a Go framework. Add Go language, or select a different framework.
Fiber (Go)	Language must include Go	Fiber is a Go framework. Add Go language, or select a different framework.
Echo (Go)	Language must include Go	Echo is a Go framework. Add Go language, or select a different framework.
Actix (Rust)	Language must include Rust	Actix is a Rust framework. Add Rust language, or select a different framework.
Axum (Rust)	Language must include Rust	Axum is a Rust framework. Add Rust language, or select a different framework.

6.2 Backend Language → Backend Framework

Field / value	Condition	Error message
TypeScript or JavaScript selected	Framework must include NestJS, Express, Fastify, Koa, Hapi, or tRPC	Selected frameworks are not compatible with TypeScript/JavaScript. Add NestJS, Express, Fastify, Koa or Hapi framework, or select Other.
Python selected	Framework must include Django, FastAPI, or Flask	Selected frameworks are not compatible with Python. Add Django, FastAPI or Flask framework, or select Other.

Field / value	Condition	Error message
Java selected	Framework must include Spring Boot	Selected frameworks are not compatible with Java. Add Spring Boot framework, or select Other.
Kotlin selected	Framework must include Spring Boot	Selected frameworks are not compatible with Kotlin. Add Spring Boot framework, or select Other.
C# selected	Framework must include ASP.NET Core	Selected frameworks are not compatible with C#. Add ASP.NET Core framework, or select Other.
PHP selected	Framework must include Laravel	Selected frameworks are not compatible with PHP. Add Laravel framework, or select Other.
Ruby selected	Framework must include Ruby on Rails	Selected frameworks are not compatible with Ruby. Add Ruby on Rails framework, or select Other.
Elixir selected	Framework must include Phoenix	Selected frameworks are not compatible with Elixir. Add Phoenix framework, or select Other.
Go selected	Framework must include Gin, Fiber, or Echo	Selected frameworks are not compatible with Go. Add Go framework (Gin, Fiber, Echo), or select Other.
Rust selected	Framework must include Actix or Axum	Selected frameworks are not compatible with Rust. Add Rust framework (Actix, Axum), or select Other.

6.3 ORM → Backend Language

Field / value	Condition	Error message
Prisma	Language must include TypeScript or JavaScript	Prisma is a TypeScript/JavaScript ORM. Add TypeScript or JavaScript language, or select a different ORM.
TypeORM	Language must include TypeScript or JavaScript	TypeORM is a TypeScript/JavaScript ORM. Add TypeScript or JavaScript

Field / value	Condition	Error message
		language, or select a different ORM.
Drizzle	Language must include TypeScript or JavaScript	Drizzle is a TypeScript/JavaScript ORM. Add TypeScript or JavaScript language, or select a different ORM.
Sequelize	Language must include TypeScript or JavaScript	Sequelize is a TypeScript/JavaScript ORM. Add TypeScript or JavaScript language, or select a different ORM.
Mongoose	Language must include TypeScript or JavaScript	Mongoose is a TypeScript/JavaScript ORM. Add TypeScript or JavaScript language, or select a different ORM.
SQLAlchemy	Language must include Python	SQLAlchemy is a Python ORM. Add Python language, or select a different ORM.
Django ORM	Language must include Python	Django ORM is a Python ORM. Add Python language, or select a different ORM.
Hibernate	Language must include Java or Kotlin	Hibernate is a JVM ORM. Add Java or Kotlin language, or select a different ORM.
Entity Framework Core	Language must include C#	Entity Framework Core is a C# ORM. Add C# language, or select a different ORM.
ActiveRecord (Rails)	Language must include Ruby	ActiveRecord is a Ruby ORM. Add Ruby language, or select a different ORM.
Ecto (Elixir)	Language must include Elixir	Ecto is an Elixir ORM. Add Elixir language, or select a different ORM.
GORM (Go)	Language must include Go	GORM is a Go ORM. Add Go language, or select a different ORM.
sqlx (Go/Rust)	Language must include Go or Rust	sqlx supports Go and Rust. Add Rust language, or select a different ORM.
Eloquent (Laravel)	Language must include PHP	Eloquent is a PHP ORM. Add PHP language, or select a different ORM.

6.4 ORM → Backend Framework (tightly coupled ORMs)

Field / value	Condition	Error message
Django ORM	Framework must include Django or Other	Django ORM is inseparable from the Django framework. Add Django framework, or select a different ORM.
ActiveRecord (Rails)	Framework must include Ruby on Rails or Other	ActiveRecord is inseparable from Ruby on Rails. Add Ruby on Rails framework, or select a different ORM.
Ecto (Elixir)	Framework must include Phoenix or Other	Ecto is inseparable from the Phoenix framework. Add Phoenix language, or select a different ORM.
Eloquent (Laravel)	Framework must include Laravel or Other	Eloquent is inseparable from the Laravel framework. Add Laravel framework, or select a different ORM.

6.5 ORM → Database

Field / value	Condition	Error message
Mongoose	Database must include MongoDB or Other	Mongoose is a MongoDB-specific ORM. Add MongoDB or select a different ORM.
Django ORM	Database must include PostgreSQL, MySQL, SQLite, or Other	Django ORM does not support document or graph databases natively. Add PostgreSQL, MySQL, SQLite, or select a different ORM.
Hibernate	Database must include a relational DB (PostgreSQL, MySQL, SQL Server, etc.), or Other	Hibernate is a relational ORM. Add relational database(s), or select a different ORM.
Entity Framework Core	Database must include a relational DB (PostgreSQL, MySQL, SQL Server, etc.), or Other	Entity Framework Core is a relational ORM. Add relational database(s), or select a different ORM.
Prisma, TypeORM	Database must include a relational DB or MongoDB, or Other	Selected ORM does not support the chosen databases natively. Add relational database(s) or MongoDB, or select a different ORM.

Field / value	Condition	Error message
Drizzle, Sequelize	Database must include a relational DB, or Other	Selected ORM does not support the chosen databases natively. Add relational database(s), or select a different ORM.