1. Introduction

This document provides a detailed specification of the XYZ API, intended for backend engineers and system integrators.

It includes architecture diagrams, data models, authentication methods, and error handling conventions.

2. Architecture Overview

2.1 System Components

2.1.1 API Gateway

The API Gateway acts as the entry point for all client requests. It handles routing, rate limiting, and request validation.

2.1.2 Application Server

The core logic resides in stateless microservices, each responsible for a domain (users, billing, notifications).

2.1.3 Database Layer

Data is persisted using a PostgreSQL instance with read replicas for horizontal scaling.

2.2 Communication Protocols

Services communicate internally using gRPC, while clients use HTTPS and REST/JSON payloads.

Event-based messaging is handled via Kafka for async communication.

3. Authentication

3.1 JWT Authentication

Each token contains user_id, scopes, and an expiration timestamp. Tokens are signed using RS256.

3.2 OAuth2 Implementation

OAuth2 flows include Authorization Code and Implicit for frontend apps, and Client Credentials for

server-to-server apps.

4. Authorization

4.1 Role-Based Access Control (RBAC)

Permissions are assigned to roles such as 'admin', 'editor', and 'viewer'.

4.2 Attribute-Based Access Control (ABAC)

Policies can also be defined based on user attributes like department or project ownership.

5. Data Models

5.1 Core Models

5.1.1 User

Contains id, name, email, is_active, roles.

5.1.2 Project

Includes id, owner_id, title, status, created_at.

5.2 Nested Models

5.2.1 AuditLog

Captures user actions including timestamp, action_type, resource_id, and metadata.

5.2.2 Settings

User-specific configuration such as notifications, themes, and privacy preferences.

6. API Endpoints

6.1 User Endpoints

GET /users - list users

POST /users - create user

6.2 Project Endpoints

GET /projects - retrieve list

POST /projects - create new project

7. Webhooks

7.1 Event Types

Events include user.created, user.deleted, project.updated, audit.logged.

7.2 Security

All payloads are signed with HMAC SHA-256 and include a timestamp to prevent replay attacks.

8. Error Handling

Errors follow a consistent schema with code, message, and optional details.

HTTP 400 - Bad Request, HTTP 401 - Unauthorized, HTTP 500 - Internal Error.

9. Deployment

9.1 CI/CD

GitHub Actions is used for continuous integration. Production deployments use Kubernetes and Helm charts.

9.2 Monitoring

Metrics are exported to Prometheus. Alerts are routed via Grafana OnCall.

10. Security

10.1 Encryption

All sensitive data is encrypted at rest with AES-256 and in transit via TLS 1.3.

10.2 Compliance

The platform complies with SOC 2 and GDPR. Data retention policies are enforced via database triggers and background jobs.

11. Versioning

11.1 SemVer

Major versions indicate breaking changes. Minor/patch versions are backward compatible.

11.2 Deprecation Policy

Deprecated features are supported for 6 months before removal. Clients are notified via dashboard alerts and changelogs.