High-Level Design (HLD) – Rule Engine Microservice

1. Introduction

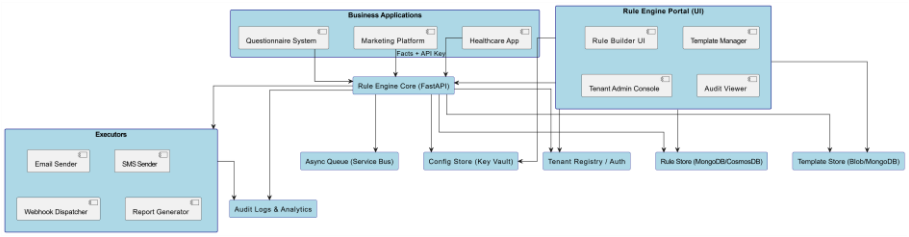
This document describes the High-Level Design (HLD) of a Rule Engine Microservice, built using Pytho

2. Scope

In Scope:
- Rule definition, storage, and evaluation.
- Multi-tenant isolation (per organisation).
- Action execution (email, SMS, webhook, reports).
- Configs & templates management.
- Audit logging and monitoring.

Out of Scope:
- Low-level system infra (VM sizing, CI/CD pipelines).
- Data ownership in source systems.
- Business apps' user management (only tenant-level mapping is supported).

3. Architecture Overview



The system is composed of:
- Rule Engine Core (FastAPI): Stateless service responsible for rule evaluation and action execution
- Rule Engine Portal (UI): Provides a business-friendly interface for tenants to configure rules, te
- Data Stores: MongoDB for rules/templates/logs, Azure App Config + Key Vault for tenant configs and
- Executors: Plugins for outbound actions like email, SMS, webhooks, reports.
- Queue: Asynchronous processing for heavy/bulk workflows.

4. Key Components

API Gateway / Rule Engine Service (FastAPI): Provides REST/GraphQL endpoints (/evaluate, /rules, /te

Rule Store (MongoDB): Stores rules in JSON format, partitioned by tenant_id.

Template Store: Stores templates for emails, SMS, or reports (MongoDB/Blob).

Config Store: Tenant-specific integration configs (Azure App Config + Key Vault).

Rule Evaluator: Parses and evaluates rules against facts.

Executor Engine: Executes actions (email, SMS, report, webhook, custom).

Audit Logs & Analytics: Stores rule evaluations and actions per tenant.

Tenant Registry: Central registry for onboarded organisations and API keys.

5. Data Flow

1. Business app sends facts + API key → /evaluate.
2. Rule engine validates API key → resolves tenant_id.
3. Fetches rules for that tenant.
4. Evaluates facts against rules.

5. Executes matching actions.
6. Loads tenant configs & templates.
7. Executes outbound action.
8. Stores audit log.

6. Security

- Tenant isolation enforced by tenant_id.
- Authentication via API keys / OAuth2.
- Authorization via RBAC in portal.
- Secrets stored in Azure Key Vault.
- HTTPS/TLS enforced.
- Immutable audit logs.

7. Scalability & Availability

- Stateless FastAPI services (scalable via Azure App Service/AKS).
- MongoDB cluster sharded by tenant.
- Async execution using Azure Service Bus / Kafka.
- CDN + caching for templates.
- Multi-region deployment.

8. Extensibility

- New executors can be added as plugins.
- New integrations configured per tenant without code changes.
- Portal supports drag-and-drop UI for non-technical users.