High-Level Design (HLD)

High-Level Design Document

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Executive Summary

Purpose: {{ONE_LINE_DESCRIPTION_OF_SYSTEM}}

Business Context: {{WHY_THIS_SYSTEM_EXISTS}}

Key Outcomes:

- {{OUTCOME_1}}
- {{OUTCOME_2}}
- {{OUTCOME_3}}

Scope: This document covers {{IN_SCOPE}} and excludes {{OUT_OF_SCOPE}}.

---

2. System Context (C4 Level 1)

```
C4Context
    title System Context — {{PROJECT_NAME}}

    Person(user, "{{PRIMARY_USER_TYPE}}", "{{USER_DESCRIPTION}}")
    Person(admin, "System Administrator", "Manages and configures the system")

    System(system, "{{SYSTEM_NAME}}", "{{SYSTEM_SHORT_DESCRIPTION}}")

    System_Ext(extSystem1, "{{EXTERNAL_SYSTEM_1}}", "{{EXT_SYSTEM_1_DESCRIPTION}}")
    System_Ext(extSystem2, "{{EXTERNAL_SYSTEM_2}}", "{{EXT_SYSTEM_2_DESCRIPTION}}")
    System_Ext(emailSystem, "Email Provider", "Sends transactional emails")

    Rel(user, system, "Uses", "HTTPS")
    Rel(admin, system, "Manages", "HTTPS")
    Rel(system, extSystem1, "Calls", "REST/HTTPS")
    Rel(system, extSystem2, "Publishes events to", "AMQP")
    Rel(system, emailSystem, "Sends emails via", "SMTP/API")
```

---

3. Container Diagram (C4 Level 2)

```
C4Container
    title Container Diagram — {{PROJECT_NAME}}

    Person(user, "{{PRIMARY_USER_TYPE}}")

    Container_Boundary(system, "{{SYSTEM_NAME}}") {
        Container(webApp, "Web Application", "{{FRONTEND_TECH}}", "Single-page application served to users")
        Container(api, "API Gateway / Backend", "{{BACKEND_TECH}}", "Handles business logic and orchestration")
        Container(workerService, "Background Worker", "{{WORKER_TECH}}", "Processes async jobs and scheduled tasks")
        ContainerDb(database, "Primary Database", "{{DB_TECH}}", "Stores persistent application data")
        ContainerDb(cache, "Cache Layer", "Redis", "Session storage and hot data caching")
        Container(messageQueue, "Message Queue", "{{QUEUE_TECH}}", "Async event bus between services")
    }
```

```
    System_Ext(extApi, "{{EXTERNAL_API}}", "Third-party integration")

    Rel(user, webApp, "Visits", "HTTPS")
    Rel(webApp, api, "Calls", "REST/HTTPS")
    Rel(api, database, "Reads/Writes", "TCP")
    Rel(api, cache, "Reads/Writes", "TCP")
    Rel(api, messageQueue, "Publishes events", "AMQP")
    Rel(workerService, messageQueue, "Consumes events", "AMQP")
    Rel(workerService, database, "Reads/Writes", "TCP")
    Rel(api, extApi, "Calls", "REST/HTTPS")
```

---

4. Component Overview

Component	Responsibility	Technology	Owner Team
{{COMPONENT_1}}	{{RESPONSIBILITY_1}}	{{TECH_1}}	{{TEAM_1}}
{{COMPONENT_2}}	{{RESPONSIBILITY_2}}	{{TECH_2}}	{{TEAM_2}}
{{COMPONENT_3}}	{{RESPONSIBILITY_3}}	{{TECH_3}}	{{TEAM_3}}

Component Descriptions

{{COMPONENT_1}}

Responsibility: {{DETAILED_RESPONSIBILITY}} **Key Interfaces:** {{INTERFACE_DESCRIPTION}}
Rationale: {{WHY_SEPARATE}}

---

5. Technology Stack

Layer	Technology	Version	Rationale
Frontend Framework	{{FE_FRAMEWORK}}	{{VERSION}}	{{RATIONALE}}
UI Component Library	{{UI_LIB}}	{{VERSION}}	{{RATIONALE}}
Backend Language	{{LANG}}	{{VERSION}}	{{RATIONALE}}
Backend Framework	{{BE_FRAMEWORK}}	{{VERSION}}	{{RATIONALE}}
Primary Database	{{DB}}	{{VERSION}}	{{RATIONALE}}

Layer	Technology	Version	Rationale
Cache	{{CACHE}}	{{VERSION}}	{{RATIONALE}}
Message Queue	{{QUEUE}}	{{VERSION}}	{{RATIONALE}}
Search Engine	{{SEARCH}}	{{VERSION}}	{{RATIONALE}}
Object Storage	{{STORAGE}}	{{VERSION}}	{{RATIONALE}}
Container Runtime	{{CONTAINER}}	{{VERSION}}	{{RATIONALE}}
Orchestration	{{ORCHESTRATION}}	{{VERSION}}	{{RATIONALE}}
API Gateway	{{GATEWAY}}	{{VERSION}}	{{RATIONALE}}
Auth Provider	{{AUTH}}	{{VERSION}}	{{RATIONALE}}
Observability	{{OBSERVABILITY}}	{{VERSION}}	{{RATIONALE}}
CI/CD	{{CICD}}	{{VERSION}}	{{RATIONALE}}

---

6. Data Flow Overview

6.1 Primary Write Flow

```
flowchart LR
    A([User]) -->|HTTPS POST| B[API Gateway]
    B -->|Authenticate| C[Auth Service]
    C -->|JWT validated| B
    B -->|Route request| D[Business Service]
    D -->|Validate input| D
    D -->|Write| E[(Database)]
    D -->|Publish event| F[Message Queue]
    F -->|Consume| G[Worker Service]
    G -->|Side effects| H[External APIs]
    G -->|Notify| I[Email/Push]
    D -->|Cache invalidate| J[(Cache)]
    D -->|Return 201| B
    B -->|Response| A
```

6.2 Primary Read Flow

```
flowchart LR
    A([User]) -->|HTTPS GET| B[API Gateway]
    B -->|Authenticate| C[Auth Service]
    B -->|Route| D[Business Service]
    D -->|Cache check| E[(Redis Cache)]
    E -->|Cache hit| D
    E -->|Cache miss| F[(Database)]
    F -->|Read| D
    D -->|Populate cache| E
    D -->|Return 200| A
```

---

7. Integration Points

7.1 External Integrations

System	Direction	Protocol	Auth	Data Exchanged	SLA/Criticality
{{EXT_SYSTEM_1}}	Outbound	REST/HTTPS	API Key	{{DATA}}	{{SLA}} / {{CRITICALITY}}
{{EXT_SYSTEM_2}}	Inbound	Webhooks	HMAC	{{DATA}}	{{SLA}} / {{CRITICALITY}}
{{EXT_SYSTEM_3}}	Bidirectional	gRPC	mTLS	{{DATA}}	{{SLA}} / {{CRITICALITY}}

7.2 Internal Service Integrations

Service	Integration Type	Protocol	Notes
{{INTERNAL_SERVICE_1}}	Synchronous	REST	{{NOTES}}
{{INTERNAL_SERVICE_2}}	Asynchronous	Events	{{NOTES}}

---

8. Deployment Overview

```
flowchart TB
    subgraph Internet
```

```
    CDN[CDN / Edge Cache]
    DNS[DNS]
end

subgraph Cloud["Cloud Provider — {{CLOUD_PROVIDER}}"]
    subgraph LoadBalancer["Load Balancer Layer"]
        LB[Application Load Balancer]
    end

    subgraph AppTier["Application Tier — {{REGION}}"]
        direction LR
        API1[API Pod 1]
        API2[API Pod 2]
        API3[API Pod N]
    end

    subgraph WorkerTier["Worker Tier"]
        W1[Worker Pod 1]
        W2[Worker Pod N]
    end

    subgraph DataTier["Data Tier"]
        DB_PRIMARY[(DB Primary)]
        DB_REPLICA[(DB Replica)]
        REDIS[(Redis Cluster)]
        MQ[Message Queue]
    end

    subgraph Observability["Observability Stack"]
        LOGS[Log Aggregator]
        METRICS[Metrics / Prometheus]
        TRACES[Distributed Tracing]
    end
end

DNS --> CDN
CDN --> LB
LB --> API1 & API2 & API3
API1 & API2 & API3 --> DB_PRIMARY
API1 & API2 & API3 --> REDIS
```

```
    API1 & API2 & API3 --> MQ
    DB_PRIMARY --> DB_REPLICA
    MQ --> W1 & W2
    API1 & API2 & API3 --> LOGS & METRICS & TRACES
```

Environments

Environment	URL	Purpose	Scale
Development	http://localhost:{{PORT}}	Local dev	Single instance
Staging	https://staging.{{DOMAIN}}	Pre-prod testing	Minimal (1 replica)
Production	https://{{DOMAIN}}	Live traffic	Auto-scaled

---

9. Cross-Cutting Concerns

9.1 Authentication & Authorization

- **Strategy:** {{AUTH_STRATEGY}} (e.g., JWT Bearer tokens / OAuth2 / Session-based)
- **Identity Provider:** {{IDP}} (e.g., Auth0, Keycloak, custom)
- **Authorization Model:** {{AUTHZ_MODEL}} (e.g., RBAC, ABAC)
- **Token Lifetime:** Access: {{ACCESS_TTL}} | Refresh: {{REFRESH_TTL}}
- **MFA:** {{MFA_REQUIRED}} — {{MFA_METHOD}}

9.2 Logging

- **Framework:** {{LOGGING_FRAMEWORK}}
- **Format:** JSON structured logs
- **Levels:** DEBUG (dev), INFO (staging/prod), WARN/ERROR (alerts)
- **Correlation IDs:** X-Request-ID header propagated across all services
- **Retention:** {{LOG_RETENTION_DAYS}} days in {{LOG_STORAGE}}
- **PII Handling:** PII fields masked/redacted before logging

9.3 Error Handling

- **API Errors:** RFC 7807 Problem Details format
- **Retry Strategy:** Exponential backoff with jitter (max {{MAX_RETRIES}} retries)
- **Circuit Breaker:** Enabled on external calls — threshold: {{CB_THRESHOLD}}% failure rate

- **Dead Letter Queue:** Failed messages → DLQ with {{DLQ_RETENTION}} retention

9.4 Caching

- **Strategy:** Cache-aside pattern
- **Cache Invalidation:** {{INVALIDATION_STRATEGY}}
- **TTLs:** Session: {{SESSION_TTL}} | API responses: {{API_CACHE_TTL}} | Reference data: {{REF_TTL}}
- **Cache Penetration Protection:** Bloom filter / null value caching

9.5 Rate Limiting

- **Implementation:** {{RATE_LIMIT_IMPLEMENTATION}} (e.g., Redis sliding window)
- **Default Limits:** {{REQUESTS_PER_MINUTE}} req/min per IP | {{AUTH_REQUESTS_PER_MINUTE}} req/min per authenticated user
- **Response:** HTTP 429 with Retry-After header

9.6 Secrets Management

- **Tool:** {{SECRETS_MANAGER}} (e.g., HashiCorp Vault, AWS Secrets Manager)
- **Rotation:** {{ROTATION_POLICY}}
- **Principle:** No secrets in code, environment files committed to VCS, or logs

---

10. Quality Attributes & Architectural Trade-offs

Quality Attribute	Target	Approach	Trade-off
Availability	{{SLA_PERCENT}} uptime	Multi-AZ deployment, health checks, auto-restart	Higher infrastructure cost
Performance (p99 latency)	< {{P99_LATENCY}}ms	Caching, query optimization, CDN	Cache invalidation complexity
Scalability	{{CONCURRENT_USERS}} concurrent users	Horizontal scaling, stateless services	Distributed state challenges
Security	OWASP Top 10 compliant	WAF, input validation, RBAC	Added latency from security checks
Maintainability	{{DEPLOY_FREQUENCY}} deploys/week	CI/CD pipeline, test coverage > {{TEST_COVERAGE}}%	Initial investment in tooling

Quality Attribute	Target	Approach	Trade-off
Data Consistency	{{CONSISTENCY_MODEL}}	{{CONSISTENCY_APPROACH}}	{{CONSISTENCY_TRADEOFF}}

11. Key Architectural Decisions

ADR	Decision	Status	Date
ADR-001	{{DECISION_SUMMARY_1}}	Accepted	{{DATE}}
ADR-002	{{DECISION_SUMMARY_2}}	Accepted	{{DATE}}
ADR-003	{{DECISION_SUMMARY_3}}	Proposed	{{DATE}}

12. Constraints & Assumptions

12.1 Constraints

#	Constraint	Category	Impact
C1	{{CONSTRAINT_1}}	Technical/Regulatory/Business	{{IMPACT}}
C2	{{CONSTRAINT_2}}	Technical/Regulatory/Business	{{IMPACT}}
C3	{{CONSTRAINT_3}}	Technical/Regulatory/Business	{{IMPACT}}

12.2 Assumptions

#	Assumption	Validation Method	Risk if Wrong
A1	{{ASSUMPTION_1}}	{{HOW_TO_VALIDATE}}	{{RISK}}
A2	{{ASSUMPTION_2}}	{{HOW_TO_VALIDATE}}	{{RISK}}

13. Risks & Mitigations

Risk	Likelihood	Impact	Score	Mitigation	Contingency
{{RISK_1}}	{{1-5}}	{{1-5}}	{{L×I}}	{{MITIGATION}}	{{CONTINGENCY}}
{{RISK_2}}	{{1-5}}	{{1-5}}	{{L×I}}	{{MITIGATION}}	{{CONTINGENCY}}
{{RISK_3}}	{{1-5}}	{{1-5}}	{{L×I}}	{{MITIGATION}}	{{CONTINGENCY}}
Single database bottleneck	3	5	15	Read replicas, connection pooling	Add read replicas, implement CQRS
Third-party API unavailability	4	3	12	Circuit breaker, cached fallback	Fallback to cached data, async retry
Data breach via injection	2	5	10	Input validation, parameterized queries, WAF	Incident response plan, GDPR notification

Approval

Role	Name	Date	Signature
Author			
Technical Lead			
Security Review			
Architect			
Approver (CTO/Lead)			

Revision #4
Created 2026-02-24 14:52:22 UTC by John
Updated 2026-05-25 07:31:46 UTC by John