Automated Validation of User Stories Against CR Documents

Technical Architecture Document (TAD) – COMPLETE EDITION

Version: 3.0

Generated On: 2026-02-25 04:55:42 UTC

Architecture Style: API-First, Modular Monolith

Execution Model: Manual On-Demand Validation

---

1. Executive Summary

This system validates one User Story at a time against selected Change Request (CR) documents and related artifacts using semantic retrieval and structured LLM validation.

It produces:

- Alignment Summary

- Functional Gaps

- Acceptance Criteria Gaps

- Business Rule Gaps

- NFR Gaps

- Risk Indicators

- Suggested Improvements

- Traceability Matrix

- Readiness Score

- Structured JSON Output

- HTML Report

---

2. Architecture Principles

- API-first design
- Stateless validation service
- Hybrid embedding model
- Full CR versioning
- Incremental ingestion via checksum
- Immutable validation core
- Human comment overlay
- Enterprise audit compliance
- Scalable without microservices
- Phased rollout ready

---

3. High-Level Architecture

```
     ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
     ■        Optional Web Dashboard          ■
     ■ (Validation Status & Risk View)        ■
     ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                        ■ REST
     ■■■■■■■■■■■■■■■■■■■■▼■■■■■■■■■■■■■■■■■■■■
     ■          API Layer (Primary)           ■
     ■  - POST /validate                      ■
     ■  - GET /validation/{id}                ■
     ■  - API Key Auth                        ■
     ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                        ■
     ■■■■■■■■■■■■■■■■■■■■▼■■■■■■■■■■■■■■■■■■■■
     ■     Validation Orchestrator            ■
     ■     (Stateless Core Engine)            ■
     ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
                     ■               ■
■■■■■■■■■■■■■■■■■■■■■■▼■■■■   ■■■■■■■■▼■■■■■■■■■■■■■■■■■■
■ Runtime Story Embedder ■    ■ Azure DevOps Connector ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■   ■■■■■■■■■■■■■■■■■■■■■■■■■■■
                ■                        ■
     ■■■■■■■■■■■▼■■■■■■■■■■■■■■■■■■■■■■■■▼■■■■■■■■■■■■■■■
     ■         Vector Retrieval Layer                 ■
     ■  (CR, Tech Docs, NFR, Defects, Releases)      ■
     ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

```
                          ■
                ■■■■■■■■■■■■▼■■■■■■■■■■■■
                ■   Context Builder    ■
                ■■■■■■■■■■■■■■■■■■■■■■■■■
                          ■
                ■■■■■■■■■■■■▼■■■■■■■■■■■■
                ■   LLM Validation     ■
                ■ (Structured Prompt) ■
                ■■■■■■■■■■■■■■■■■■■■■■■■■
                          ■
                ■■■■■■■■■■■■▼■■■■■■■■■■■■
                ■  Scoring Engine      ■
                ■■■■■■■■■■■■■■■■■■■■■■■■■
                          ■
        ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
        ■                          ■                          ■
■■■■■■■■■■▼■■■■■■■■■■     ■■■■■■■■■■■■▼■■■■■■■■■■     ■■■■■■■■■■■■▼■■■■■■■■■■■■
■ Relational DB   ■      ■     Vector DB     ■      ■ HTML File Storage  ■
■ Reports & Audit ■      ■  CR Embeddings    ■      ■ Generated Reports  ■
■■■■■■■■■■■■■■■■■■■■■     ■■■■■■■■■■■■■■■■■■■■■■■     ■■■■■■■■■■■■■■■■■■■■■■■■
```

■■

■ Optional Web Dashboard ■

■ (Validation Status & Risk View) ■

■■

■ REST

■■■■■■■■■■■■■■■■■■■■■▼■■■■■■■■■■■■■■■■■■■■■

■ API Layer (Primary) ■

■ - POST /validate ■

■ - GET /validation/{id} ■

■ - API Key Auth ■

■■

■

■■■■■■■■■■■■■■■■■■■■■▼■■■■■■■■■■■■■■■■■■■■■

■ Validation Orchestrator ■

■ (Stateless Core Engine) ■

■■

■ ■

■■■■■■■■■■■■■■■■■■■■■■■■▼■■■■ ■■■■■■■■■▼■■■■■■■■■■■■■■■■■■

■ Runtime Story Embedder ■ ■ Azure DevOps Connector ■

```
■■■■■■■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■■■■■■■

■ ■

■■■■■■■■■■■▼■■■■■■■■■■■■■■■■■■■■■■■▼■■■■■■■■■■■■■■

■ Vector Retrieval Layer ■

■ (CR, Tech Docs, NFR, Defects, Releases) ■

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

■

■■■■■■■■■■■▼■■■■■■■■■■■

■ Context Builder ■

■■■■■■■■■■■■■■■■■■■■■■■■

■

■■■■■■■■■■■▼■■■■■■■■■■■

■ LLM Validation ■

■ (Structured Prompt) ■

■■■■■■■■■■■■■■■■■■■■■■■■

■

■■■■■■■■■■■▼■■■■■■■■■■■

■ Scoring Engine ■

■■■■■■■■■■■■■■■■■■■■■■■■

■

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

■ ■ ■

■■■■■■■■■▼■■■■■■■■■ ■■■■■■■■■■■▼■■■■■■■■■
■■■■■■■■■■■▼■■■■■■■■■■

■ Relational DB ■ ■ Vector DB ■ ■ HTML File Storage ■

■ Reports & Audit ■ ■ CR Embeddings ■ ■ Generated Reports ■

■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■■■
■■■■■■■■■■■■■■■■■■■■■■■
```

---

4. Component Breakdown

4.1 API Layer (Primary)

Endpoints:

- POST /api/v1/validate
- GET /api/v1/validation/{id}
- GET /api/v1/health

Authentication:

- API Key
- Rate limiting
- Request logging

4.2 Validation Orchestrator

Responsibilities:

- Retrieve story from Azure DevOps
- Embed story at runtime
- Retrieve CR context from Vector DB
- Expand semantic context
- Execute structured LLM prompt
- Compute scoring
- Persist validation
- Generate HTML report

Stateless and horizontally scalable.

---

5. AI / LLM Design

5.1 Embedding Strategy

- CR Documents → Pre-embedded
- Tech Docs → Pre-embedded
- NFR Docs → Pre-embedded
- Release Notes → Pre-embedded
- Defects → Pre-embedded
- User Story → Runtime embedding

5.2 Retrieval Flow

1. Embed story
2. Filter by selected CR IDs
3. Top-K semantic retrieval
4. Context expansion
5. Structured validation

5.3 Prompt Phases

1. Functional Alignment
2. Acceptance Criteria Gap Detection
3. Business Rule Validation
4. NFR Validation
5. Ambiguity Detection
6. Risk Classification
7. Readiness Scoring
8. Evidence Citation Enforcement

---

6. Validation Flow Phases

Phase 1 – Story Retrieval

Phase 2 – CR Semantic Retrieval

Phase 3 – Context Aggregation

Phase 4 – Structured Prompt Execution

Phase 5 – Gap Detection & Scoring

Phase 6 – Report Generation

Phase 7 – Storage & Audit Logging

---

7. Scoring Model

Dimensions:

- Functional Alignment (25%)

- Acceptance Criteria (15%)

- Business Rules (15%)

- NFR (15%)

- Ambiguity (10%)

- Risk (10%)

- Traceability (10%)

Risk Bands:

- LOW (80–100)

- MEDIUM (60–79)

- HIGH (<60)

---

8. Database Design (Second Layer)

validation_reports

- validation_id
- story_id
- project_id
- readiness_score
- risk_level
- prompt_version
- model_version
- cr_versions_used
- html_path
- created_at

validation_scores

- validation_id
- dimension
- score
- rationale

validation_comments

- comment_id
- validation_id
- user_id
- comment_text
- timestamp

audit_log

- event_type

- metadata

- timestamp

---

9. Vector DB Schema

Each chunk includes:

- documentId

- version

- sectionId

- projectId

- sourceType

- checksum

- embeddingModelVersion

- indexedTimestamp

Supports full re-index tracking and audit reproducibility.

---

10. HTML Report Structure

- Executive Summary

- Alignment Summary

- Functional Gaps

- AC Gaps

- Business Rule Gaps

- NFR Gaps

- Risk Indicators

- Suggested Improvements

- Traceability Matrix

- Evidence Citations

- Readiness Score Dashboard

File Naming:

StoryId_ValidationId_Timestamp.html

---

11. Deployment Model

Phase 1:

- Single containerized service

- Managed LLM API

- External Vector DB

- Relational DB

- File storage

Phase 2:

- Horizontal scaling

- Worker queues

- Distributed ingestion

---

12. Scaling Strategy

- Stateless API nodes

- Separate ingestion worker

- Vector DB scaling

- Read replica DB support

---

13. Monitoring & Observability

Metrics:

- API latency
- LLM token usage
- Retrieval latency
- Validation duration
- Error rate

Audit Captured:

- Prompt version
- Model version
- CR versions
- Retrieval chunk IDs
- Score breakdown

---

14. Security

- API Key authentication
- TLS encryption
- Secret management
- Immutable validation records
- Full audit logs

---

15. Risk Mitigation

- LLM hallucination → Citation enforcement

- Version drift → Version-tagged retrieval
- Scalability risk → Stateless design
- Cost overrun → Token monitoring
- Over-engineering → Modular monolith

---

16. Future Extensibility

- OAuth2 integration
- Configurable scoring
- Azure DevOps write-back
- Multi-tenant isolation
- Predictive defect modeling
- Executive analytics dashboard

---

17. Conclusion

This architecture:

- Is API-first
- Uses DB as second layer
- Keeps Web layer optional
- Provides audit-grade explainability
- Enables scalable RAG-based validation
- Avoids unnecessary microservices
- Supports phased enterprise rollout
- Ensures clarity, traceability, and maintainability