Color Value Validation Requirements

Evaluation Criteria for Web Applications in Software as a Medical Device (SaMD)

This document outlines key evaluation parameters for web applications used in medical devices, ensuring compliance, security, performance, and usability. The validation process is divided into Internal and External evaluation parameters.

Internal Evaluation Parameters

1. Technical Performance and Reliability

What it is: Ensures the system runs smoothly, responds quickly, and can handle different loads without failure.

How to test:

System response time and latency measurements: Use tools like Google Lighthouse, Pingdom, or WebPageTest to measure response times.

Server uptime and availability metrics: Use UptimeRobot or AWS CloudWatch for real-time monitoring.

Database performance and data integrity checks: Run SQL queries for consistency and integrity checks. Use database profiling tools like MySQL Performance Schema.

Load testing results and scalability metrics: Use JMeter or LoadRunner to simulate thousands of concurrent users.

Error handling and recovery mechanisms: Force error scenarios (e.g., wrong inputs, network failure) and validate how the system recovers.

Browser compatibility across different platforms: Test on Chrome, Firefox, Safari, Edge using tools like BrowserStack or LambdaTest.

2. Security Controls

What it is: Ensures the application protects sensitive patient data from unauthorized access.

How to test:

Authentication and authorization mechanisms: Test login security, multi-factor authentication (MFA), and role-based access control (RBAC).

Data encryption standards: Verify that TLS 1.2/1.3 is used for transit encryption and AES-256 for data at rest.

Audit trail implementation: Check if all user activities are logged and cannot be altered.

Session management security: Validate automatic logout, session timeout, and session hijacking prevention.

API security measures: Perform OWASP API Security Top 10 testing.

Vulnerability assessment results: Run scans using Nessus, Qualys, or OpenVAS.

Penetration testing outcomes: Conduct ethical hacking tests using Burp Suite, Metasploit, or Kali Linux.

3. Data Management

What it is: Ensures data is accurate, stored securely, and can be recovered if needed.

How to test:

Data validation processes: Input invalid, incorrect, or malicious data and check if the system rejects it properly.

Backup and recovery procedures: Simulate data loss and verify the recovery process.

Data integrity checks: Run checksum validation or compare pre and post-operation data.

Storage compliance with medical data regulations: Ensure compliance with HIPAA, GDPR, or FDA.

Data versioning and archival systems: Validate historical data retrieval and rollback functions.

Patient data privacy controls: Conduct privacy impact assessments and ensure access is restricted.

4. Clinical Validation

What it is: Ensures the system accurately processes medical data and provides clinically valid results.

How to test:

Algorithm accuracy and precision metrics: Run test cases against known medical datasets.

Clinical performance indicators: Compare system-generated values against gold-standard clinical values.

Validation protocols and results: Implement peer reviews, A/B testing, and retrospective analysis.

Risk assessment outcomes: Ensure potential risks are identified and mitigated.

Clinical trial data if applicable: Validate software predictions against clinical trial results.

Verification of medical calculations: Test formulas using controlled inputs and compare against expected outputs.

External Evaluation Parameters

5. Regulatory Compliance

What it is: Ensures the application meets legal and medical industry standards.

How to test:

FDA requirements (if applicable): Verify compliance with 21 CFR Part 820 and Part 11 (electronic records/signatures).

CE marking requirements: Ensure software is ISO 13485-compliant.

HIPAA compliance: Validate encryption, access controls, and data protection policies.

GDPR compliance: Ensure data minimization, user consent, and the right to be forgotten are implemented.

Regional medical device regulations: Verify compliance with country-specific rules.

Documentation requirements: Ensure proper documentation for audits and compliance checks.

Quality Management System (QMS) compliance: Ensure adherence to ISO 13485 and IEC 62304 for medical software.

6. User Interface and Experience (UI/UX)

What it is: Ensures the application is user-friendly, accessible, and prevents errors.

How to test:

Accessibility compliance (WCAG guidelines): Use WAVE or axe to check compliance.

User interface consistency: Perform manual and automated UI tests using Selenium or Cypress.

Error prevention mechanisms: Check for input validation, error messages, and data loss prevention.

Clear warning systems: Validate that alerts are noticeable and understandable.

User feedback collection: Implement user surveys, feedback tools, and A/B testing.

Documentation and help system quality: Test user guides, FAQs, and interactive tutorials.

7. Integration Capabilities

What it is: Ensures compatibility with other medical software.

How to test:

Interoperability with other medical systems: Test API integrations using Postman or SOAP UI.

HL7/FHIR compliance: Validate HL7 messages using HL7 Inspector.

EHR/EMR integration capabilities: Conduct integration tests with EPIC, Cerner, or Meditech.

API documentation and standards: Ensure APIs follow OpenAPI Specification (OAS).

Data exchange protocols: Validate JSON, XML, and HL7 standards.

8. Risk Management

What it is: Identifies and mitigates risks in the software.

How to test:

Risk assessment documentation: Review FMEA (Failure Mode and Effects Analysis) reports.

Hazard analysis: Identify system weaknesses and define mitigation strategies.

Failure mode and effects analysis (FMEA): Test system failures and measure impact.

Mitigation strategies: Ensure fallback mechanisms exist for high-risk failures.

Post-market surveillance plan: Set up real-world monitoring after release.

Incident reporting mechanisms: Verify that errors and security incidents are logged and reported properly.

9. Performance Monitoring

What it is: Ensures continuous tracking of the system’s health and efficiency.

How to test:

Real-time monitoring capabilities: Use New Relic or DataDog.

Error tracking and reporting: Implement Sentry or LogRocket.

Usage analytics: Monitor using Google Analytics or Mixpanel.

Performance metrics tracking: Use Prometheus or Grafana.

System health monitoring: Monitor CPU, memory, and network usage.

Automated alerting systems: Set up alerts for downtime, errors, or unusual activity.