National Software Development Guideline

April 2026

1.0. G National Software Development Guideline **Title**

2.0. This Guideline establishes a unified framework for the development of software systems used within Nigerian government institutions. It provides minimum requirements to ensure that software is secure, reliable, interoperable, and aligned with national standards. **Explanatory Note**

The Guideline responds to the increasing reliance on digital systems for public service delivery and the need to mitigate risks associated with poorly designed, insecure, or non-compliant software. It promotes structured development practices, adequate documentation, and the integration of security and quality considerations throughout the software development lifecycle.

The provisions of this Guideline are intended to support consistency across government systems, strengthen trust in digital services, and enable the development of sustainable and maintainable software solutions within the national digital ecosystem.

3.0. This guideline establishes the minimum requirements for the development of software to be used by Nigerian government entities. It ensures that all software meets quality, security, and operational standards, promotes the growth of the local software testing market, and enhances the efficiency and effectiveness of government services. **Objective**

4.0. This Guideline is issued under the authority provided to NITDA by the National Information Technology Development Act of 2007, which empowers NITDA to regulate the development and deployment of IT systems, including software, in Nigeria. **Authority**

5.0. This guideline applies to all software developed or modified for use by Nigerian governmental institutions, including applications, systems, and platforms that interface with government operations or services. **Scope**

6.0. This Guideline shall be read together with the Software Testing Guideline and the Software Testing Organisations Licensing Framework, which collectively constitute the National Software Quality Assurance Framework for Nigeria. **Framework Integration Clause**

7.0. The title for this guideline is National Software Development Guideline and shall come into effect when issued by NITDA. **Commencement**

8.0. For the purposes of this Guideline, the following terms shall have the meanings assigned to them below: **Definitions**

"NITDA" means the National Information Technology Development Agency, the regulatory authority responsible for the development, regulation, and oversight of Information Technology practices in Nigeria.

"Software" means any application, system, platform, or digital solution developed, deployed, or maintained for use within government operations.

"Software Development Lifecycle (SDLC)" means the structured process for developing software, including requirements, design, development, testing, deployment, and maintenance phases.

"Developer" means any individual or organisation responsible for designing, building, or maintaining software systems.

"Government Systems" means all software applications, platforms, and systems used by Nigerian government institutions.

"Risk" means the potential for adverse impact arising from system failure, security breach, data compromise, or operational disruption.

"Compliance" means adherence to the requirements, standards, and provisions set out in this Guideline and applicable laws.

"Interoperability" means the ability of software systems to exchange data and function seamlessly with other systems.

"API" means Application Programming Interface, a set of protocols and tools for building and integrating software applications.

9.0. **Risk-Based Application of Requirements**

The requirements set out in this Guideline shall be applied based on the risk profile of the software system.
Risk assessment shall consider factors including, but not limited to:

- Sensitivity of data processed
- Exposure to external networks or public access
- Criticality of the system to government operations
- Potential impact of security breaches or system failure

Software systems identified as high-risk shall be subject to enhanced requirements, including stricter security controls, comprehensive testing, and detailed documentation.

Notwithstanding the above, all software systems shall comply with the minimum requirements set out in this Guideline.
Where applicable, risk assessment shall align with national data classification and cybersecurity frameworks.

10.0. **Objectives**

- Ensure software is fit for purpose, meeting functional and non-functional requirements.
- Protect government institutions from operational risks through security, reliability, and performance standards.

11.0. **Responsibilities of Stakeholders**

NITDA

· Provide regulatory oversight for software development standards and compliance.
· Monitor adherence to this Guideline across government institutions.
· Periodically review and update the Guideline.

Government Institutions (MDAs)

· Ensure that all software developed or procured complies with this Guideline.
· Enforce compliance by vendors and internal development teams.
· Maintain documentation and records of software systems.

Software Developers and Vendors

· Develop software in accordance with the requirements of this Guideline.
· Ensure adherence to security, quality, and interoperability standards.
· Provide required documentation and support for compliance verification.

12.0. All software developed or modified for government use shall follow a structured Software Development Lifecycle (SDLC) to ensure quality, security, and maintainability.

Software Development Lifecycle (SDLC) Requirements

Requirements Phase

· Clearly define functional and non-functional requirements.

· Align requirements with business objectives and regulatory obligations.

Design Phase

· Develop system architecture and design specifications.

· Ensure security, scalability, and interoperability considerations are incorporated.

Development Phase

· Adhere to coding standards and best practices.

· Implement version control and code management processes.

Testing Phase

Testing activities shall be conducted in accordance with the Software Testing Guideline issued by NITDA.

Conduct functional, performance, security, and user acceptance testing.

· Ensure defects are identified and resolved prior to deployment.

Deployment Phase

· Follow controlled deployment procedures.

· Ensure proper configuration and environment setup.

Maintenance Phase

· Provide ongoing updates, patches, and support.

· Monitor system performance and security continuously.

Agile and Iterative Development Approaches

Software development under this Guideline may adopt Agile, iterative, or hybrid methodologies to enhance flexibility, responsiveness, and continuous delivery.

Notwithstanding the use of Agile methodologies, all phases of the Software Development Lifecycle must be adequately addressed and documented.

Agile practices shall ensure:

· Continuous integration and testing of software components.

· Incremental delivery of functional software.

· Regular stakeholder engagement and feedback.

· Maintenance of documentation to reflect evolving system requirements and design.

13.0. **Pre-Deployment Requirements**

All software systems shall undergo appropriate validation to ensure compliance with this Guideline prior to deployment as defined in the testing Guidelines.
Software shall not be deployed for government use unless it meets the required standards for quality, security, and operational readiness.

14.0. **Documentation Requirements**

All software and applications developed or modified for government use must conform to the following documentation requirements:

General Documentation Requirements
Each software must be accompanied by the following set of documentation to ensure transparency, maintainability, and effective use.

Software Requirements Specifications (SRS)

- **Functional Requirements**: Define the intended functional capabilities of the software in line with ISO/IEC/IEEE 29148:2018.
- **Non-Functional Requirements**: Specify performance, security, usability, reliability, maintainability, portability, efficiency, and other applicable non-functional requirements of the system.

Architecture Documentation

- **System Overview**: Provide a high-level understanding of the software's structure and purpose.
- **Component Diagrams**: Visualize the major components and their interactions.
- **Data Flow Diagrams**: Illustrate data movement within the system, including data processing and storage.
- **Design Decisions**: Document key architectural decisions and the rationale behind them.
- **Technology Stack**: Specify the technologies used, including programming languages, frameworks, and tools.

API Documentation

- **Endpoint Descriptions**: Provide detailed descriptions of available API endpoints for developers to interact with the software.

- **Authentication and Authorization**: Explain the mechanisms for secure access to the API.
- **Error Handling**: Document error reporting and handling mechanisms.
- **Rate Limiting**: Include information on rate limits to prevent abuse of the API.

User Documentation

- **User Guide**: A comprehensive guide to using the software's features.
- **FAQ Section**: Answers to frequently asked questions.
- **Troubleshooting**: Guidance for solving common problems.
- **Feature Walkthroughs**: Instructions for using key features.
- **Glossary**: Definitions of key terms.

Developer Documentation

- **Code Structure Overview**: Describe the codebase organization.
- **Coding Standards and Conventions**: Outline coding standards and guidelines.
- **Setup and Installation Guide**: Instructions for setting up the software.
- **Build and Deployment Instructions**: Steps for building and deploying the software.
- **Dependency Management**: Details on managing software dependencies.
- **Extensibility Guidelines**: Instructions for adding features without disrupting functionality.
- **Testing Instructions**: Provide procedures for testing the software.

Configuration Documentation

- **Configuration Files Overview**: Describe configuration files and their role.
- **Environment Variables**: List and explain environment variables.
- **Configuration Options**: Document available options for configuring the software.
- **Setup Scenarios**: Provide sample setup scenarios for different use cases.

Release Notes and Changelog:

- **Version History:** Document version changes and history.
- **Features and Bug Fixes:** List new features and resolved bugs for each release.

- **Deprecated Features:** Highlight features no longer supported.

Security Documentation

- **Security Architecture**: Detail the security architecture, covering key security mechanisms and controls.
- **Security Policies**: Outline the policies governing security operations.
- **Vulnerability Management**: Explain how vulnerabilities are identified, tracked, and resolved.
- **Penetration Testing Reports**: Include penetration testing results to verify the software's security.
- **User Access Control**: Define user roles and access levels, including how authorization is managed.

Operational Documentation

- **Deployment Procedures**: Detailed steps for deploying the software in production environments.
- **Monitoring and Alerts**: Document monitoring mechanisms and alert handling processes.
- **Backup and Recovery**: Procedures for backup and restoration of data in case of system failure.
- **Disaster Recovery Plan**: Comprehensive disaster recovery procedures.
- **Maintenance Tasks**: Routine maintenance activities required to keep the software operational.

Licensing and Legal Documentation

- **License Agreement**: Legal terms for using the software.
- **Third-Party Licenses**: List of licenses for third-party libraries and dependencies.
- **Copyright Information**: Copyright notices and intellectual property statements.
- **Compliance Documentation**: Ensure the software meets legal and regulatory compliance requirements.

15.0. Additional Software Requirements

In addition to the general requirements set out in this Guideline, software systems shall comply with any additional requirements applicable to their specific sector, use case, or operational context.

Such additional requirements may arise from sector-specific regulations, operational risks, or the unique characteristics of the software system, and shall be addressed as part of the software development lifecycle.

Sector-Specific Requirementss
Software developed for use in regulated sectors, including but not limited to finance, healthcare, education, and critical national infrastructure, shall comply with applicable sectoral standards, policies, and regulatory directives issued by the relevant authorities.

Security and Risk Considerations
Where software systems present elevated security risks due to data sensitivity, public exposure, or system criticality, additional security controls, monitoring mechanisms, and assurance processes shall be implemented.

Integration and Interoperability Context
Software designed to integrate with existing government platforms or external systems shall comply with additional interface, data exchange, and interoperability requirements necessary to ensure seamless and secure integration.

Emerging Technologies and Special Use Cases
Software systems leveraging emerging technologies, including but not limited to artificial intelligence, distributed systems, and cloud-native architectures, shall incorporate additional safeguards, transparency measures, and governance controls appropriate to their use.

Regulatory Alignment
Where multiple regulatory or policy frameworks apply, software systems shall ensure alignment and compliance across all relevant instruments, including data protection, cybersecurity, and digital service standards.
NITDA may, from time to time, issue supplementary guidance or directives to address additional requirements arising from sector developments, technological advancements, or national priorities.

OWASP Compliance **Security**

- All software must be developed in accordance with secure coding practices that mitigate the OWASP Top 10 security vulnerabilities. Compliance with the OWASP Application Security Verification Standard (ASVS) is also required to ensure the implementation of comprehensive and effective security measures.

Audit Trails

- Detailed Audit Trails: Implement logging mechanisms to capture all user actions and system events.
- Tamper-Evident Logs: Apply cryptographic techniques to ensure the integrity of logs and prevent unauthorized modifications.

Sector-Specific Security Requirements

- Industry-Specific Compliance: Software must meet the security requirements of the specific sector (e.g., healthcare, finance) in which it is to be used.

16.0. Accessibility **Usability Standards**

- **WCAG 2.1 Compliance**: Ensure software meets the Web Content Accessibility Guidelines (WCAG) 2.1, making it accessible to users of all classes, including those with disabilities.

User Experience

- **Intuitive Design**: Software should be designed with a focus on user experience, ensuring that it is intuitive and easy to navigate.
- **Feedback Mechanisms**: Incorporate mechanisms for users to provide feedback on usability and functionality, promoting continuous improvement.

17.0. Data Exchange Standards **Interoperability Standards**

- Standardized Data Formats: Ensure software uses standardized formats (JSON, XML, CSV) for data exchanges to ensure compatibility across different systems.
- National Data Exchange Regulation Compliance: Ensure compliance with national regulations governing secure and efficient data exchange.

Integration Standards

- API Compatibility: Software must support integration with existing government systems through well-defined APIs, ensuring seamless data exchange and interoperability.

18.0. Ensure that the software complies with all relevant local and international standards and regulations. This includes, but is not limited to, adhering to industry-specific guidelines, data protection laws, security standards, and any other regulatory requirements that apply to the software's functionality, development, and deployment. **Regulatory Compliance**

19.0. All software developed or deployed for use within Nigerian government institutions shall comply with the provisions of this Guideline. **Compliance and Enforcement**

NITDA shall monitor compliance and may conduct assessments, audits, or reviews to ensure adherence.

Compliance with this Guideline shall form part of the requirements for IT Project Clearance for all government software systems

Any software system that does not meet the requirements of this Guideline may be subject to review, remediation, or restriction from deployment within government systems.

20.0. Software systems shall be continuously monitored to ensure ongoing compliance with performance, security, and operational standards. **Monitoring and Continuous Improvement**
Feedback mechanisms shall be established to support continuous improvement and enhancement of software systems.

This Instrument was signed this _____ of April 2026

_____

Kashifu Inuwa Abdullahi *CCIE*

Director-General/CEO