THE UNITED REPUBLIC OF TANZANIA

PRESIDENT'S OFFICE, PUBLIC SERVICE MANAGEMENT AND GOOD GOVERNANCE

e-GOVERNMENT AUTHORITY

ISO 9001:2015 Certified

Document Name

Government Software Applications Quality Assurance Guidelines and Checklist

Document Title

eGA/EXT/APA/002

APPROVAL	Name	Job Title/ Role	Signature	Date
Approved by	Eng. Benedict B. Ndomba	Director General	[illegible]	14/05/2024

PREFACE

Since the world is moving into the era of automation, mobile apps, and post-pandemic life, reliance on software and applications continues to increase. Building quality software means solving existing problems without creating new problems that can be done through a proper management of Application Quality.

Software quality assurance (SQA) is a process which assures that all software engineering processes, methods, activities and work items are monitored and comply against the defined standards such as Standards for Development, Acquisition, Operations and Maintenance of e-Government Applications developed by the e-Government Authority. SQA incorporates all software development processes starting from defining requirements to coding until release.

The Application Quality Assurance Guidelines and Checklist was intended to ensure e-Government applications developed by Public Institutions adhere to development practices that promote quality solutions.

Quality Assurance activities include facilitation, review and internal control checking. Facilitation refers to sessions that provide guidance, clarity in understanding, mentoring and implementation support of processes/procedures.

[illegible]

Eng. Benedict B. Ndomba
DIRECTOR GENERAL

Table of Contents

PREFACE....1

1. INTRODUCTION....3
1.1. Overview....3
1.2. Purpose....3
1.3. Rationale....3
1.4. Scope....3

2. APPLICATION QUALITY ASSURANCE GUIDELINES AND CHECKLISTS....4
2.1. Guidelines....4
2.1.1. Development Framework....4
2.1.2. Development IDE....4
2.1.3. Decoupling Business Logic from the Presentation Layer....4
2.1.4. Certificates / Environment Software....5
2.1.5. Database Design....5
2.1.6. Modularized Code with No Duplication....5
2.1.7. Consistency of Code....5
2.1.8. Code Comments....6
2.1.9. Hard-Coded Values....6
2.1.10. Source Code....6
2.1.11. Field Validations....7
2.1.12. Dates....7
2.1.13. Passwords....8
2.1.14. Logging and Auditing....8
2.1.15. Error Handling – End User....8
2.1.16. Error Logging....9
2.1.17. Record Locking / Concurrent Users....10
2.1.18. System Testing....10
2.1.19. Regression Testing....10
2.1.20. Load Testing/Volume Testing....10
2.1.21. Business Requirements – Traceability....11
2.1.22. Transition To Support Personnel....11
2.2. Checklists....12

3. IMPLEMENTATION, REVIEW AND ENFORCEMENT....19

4. GLOSSARY AND ACRONYMS....19

4.1. GLOSSARY....19

4.2. ACRONYMS....19

5. RELATED DOCUMENT....19

6. DOCUMENT CONTROL....20

1. INTRODUCTION

1.1. Overview

The e-Government Authority (e-GA) is a public institution established by e-Government Act No. 10 of 2019 with the mandate to coordinate, oversee, promote e-Government initiatives as well as enforcing e-Government related policies, laws, regulations, standards and guidelines to public institutions. The Act empowers the Authority to effectively formulate, manage and enforce public institutions compliance with e-Government standards and guidelines.

Quality Assurance activities include facilitation, review and internal control checking. Facilitation refers to sessions that provide guidance, clarity in understanding, mentoring and implementation support of processes/procedures. Review and internal control checking refer to verification of the processes being implemented against plans, standards, procedures, guidelines, commitments etc.

1.2. Purpose

The guidelines and checklist assist to ensure effective and efficient quality assurance of processes in an application development life cycle through facilitation, review and internal control checking.

1.3. Rationale

These guidelines and checklist are intended to ensure custom-built applications adhere to development practices that promote quality solutions.

1.4. Scope

This document applies to software developers, quality assurance personnel, all Heads of ICT and/or ICT Project Managers who are responsible for ensuring Quality Assurance of developed e-Government Applications before they are launched into production. It also applies to the ICT Project team members during the design stage to ensure the developed application meets the quality standards at the execution stage.

2. APPLICATION QUALITY ASSURANCE GUIDELINES AND CHECKLISTS

2.1. Guidelines

2.1.1. Development Framework

2.1.1.1. Applications shall be developed using the most recent e-Government Interoperability Framework (eGIF) and Application Architecture as stated in e-Government Related Standards and Guidelines (https://www.ega.go.tz/standards). Other framework versions pose compatibility, security, and upgrade risks for Application Services.

2.1.2. Development IDE

2.1.2.1. Applications shall be developed using Integrated Development Environment (IDE) as per the Integration Architecture. This will allow Application Services resources to build and debug source code as needed.

2.1.3. Decoupling Business Logic from the Presentation Layer

2.1.3.1. Whenever possible, developers shall avoid using business logic in the presentation layer. The presentation layer should mainly be used for navigation throughout the application and presenting data to the user. For example, the use of Java code directly within JSP pages (i.e., Scriptlets) should be avoided. The preferred approach would be to use Tag Libraries (JSTL/EL); and

2.1.3.2. Presentation Layer of Web applications shall be developed using prevailing industry standards (e.g., using Stylesheets to position and control presentation elements, using relative positioning instead of absolute positioning, etc.).

2.1.4.Certificates / Environment Software

2.1.4.1. Any certificates or special software that needs to be installed on a server stack for an application to function (e.g. virus scanning software, SSL Certificates, etc.) shall be documented in the Operations Procedure Manual. Documentation should include the relevant expiration dates and the processes that must be followed for renewal; and

2.1.4.2. Application deployments in production environments shall not be comprised of any trial versions of software. All proprietary and copyrighted software should be properly licensed for Government use.

2.1.5.Database Design

2.1.5.1. Industry best practices shall be followed in the design of databases for production applications: i.e., tables normalized, exceptions documented, constraints enforced, and required fields completed (nulls not permitted); and

2.1.5.2. Each table shall have its own sequence if table keys are based on sequence numbers.

2.1.6.Modularized Code with No Duplication

2.1.6.1. As much as possible, code shall be organized into small, separate modules to avoid code duplication and to make future code changes easier to implement.

2.1.7.Consistency of Code

2.1.7.1. Code sections with similar functionality shall be written in a clear, predicable, and consistent way. Using different approaches to achieve the same basic purposes should be avoided. Project teams consisting of multiple developers should ensure that the developers follow the same coding style and naming conventions.

2.1.8.Code Comments

2.1.8.1. Code sections shall be well documented with comments. At a minimum, each section of code (code unit) should have an introductory brief and accurate description to explain the code functionality; and

2.1.8.2. Any potentially confusing / non-intuitive sections of code shall be commented thoroughly.

2.1.9.Hard-Coded Values

2.1.9.1. Hard coding of server names, database names, domain names, IP addresses, etc. within application code shall be avoided. These values should be contained in a single configuration file or database that is not part of the application build, so that it can be easily maintained for different server environments (development, testing/staging, and production) and will not need to be modified when new changes are built and deployed; and

2.1.9.2. Fixed values that are repeatedly used throughout application code shall be declared in a single location and referenced appropriately, as needed, within the application. As a practical guide, a change to one of these values should occur within a single reference point.

2.1.10. Source Code

2.1.10.1. Reusable source codes of the application shall be updated in Knowledge base repository; and

2.1.10.2. The final approved version of the Application Code shall be provided to Application Services for use and maintenance during the Transition Period.

2.1.11. Field Validations

2.1.11.1. Where applicable, validations shall be performed on both the presentation layer and the business layer. In Java, for example, validations may be done using JavaScript within JSP pages (presentation layer) but shall also be done within Java classes on the business layer; and

2.1.11.2. Validations shall be performed in such a way that they cannot be bypassed by end-users (e.g., by disabling JavaScript). Field lengths and types within an application should be consistent with the column lengths and types declared within the underlying database tables. User inputs must be sanitized (Data Validation Strategies). For more information are on specific guidelines, please reference e-Government Interoperability Framework (eGIF) and Information Architecture.

2.1.12. Dates

2.1.12.1. Dates shall be validated in the context of the established business rules of the application (e.g., given a person's birth date, is he/she eligible to vote?);

2.1.12.2. Dates shall include a timestamp when they are recorded in a database or log, and not just the month, day, and year. Timestamps will not be required in specific situations (such as a birth date field) where a timestamp does not make sense; and

2.1.12.3. The testers shall use date values that occurred in the past, on the target date, and in the future when testing functionality that is built around date checks.

2.1.13. Passwords

2.1.13.1. Appropriate password management shall be adhered in order to maintain the confidentiality, integrity, availability of the data maintained by the software application and reduce the risk of inappropriate access and use.

2.1.14. Logging and Auditing

2.1.14.1. The Application Architecture and ICT Security Architecture documents the functional controls related to Logging and Auditing that must be in place to adequately protect an ICT asset (application). Functional control requirements will increase as the sensitivity of the data contained in the application increases.

2.1.15. Error Handling – End User

2.1.15.1. Error messages presented to the end user shall contain only that information which will allow the user to take corrective action (e.g., "Invalid date, please re-enter in YYYY-MM-DD format");

2.1.15.2. In the case of unhandled exceptions, messages shall be generic. Avoid displaying system information in error messages such as server names, versions, and patch information, as well as application variables, paths, and other configuration information. Avoid messages that could potentially lead to system exploitation e.g., "Incorrect Username or Password" is acceptable while the message "Incorrect Password" is not;

2.1.15.3. Error handling logic shall be robust enough to gracefully and meaningfully handle all errors which could be reasonably expected to occur from user interactions with the system; and

2.1.15.4. The text for error messages shall be contained in a single location within the application code or database to facilitate quick additions and modifications by support staff.

2.1.16. Error Logging

2.1.16.1. Application errors shall be logged for support personnel in database tables that will be directly accessible to Application Services personnel. SQL can then be used to aid in searches for specific errors;

2.1.16.2. Log files for individual server tiers (i.e., Web and Application tiers) shall be kept in a single directory on each server. Also, log files shall be saved on daily basis with a time-date stamp on each file.

2.1.16.3. The error messages that are logged shall contain information that is useful for support personnel (no sensitive or personal data), such as which module of code encountered the error and what the specific error was. Meaningful and detailed error messages are particularly important when troubleshooting unknown/unexpected errors. These shall be captured and logged; and

2.1.16.4. Logging logic within applications shall be written in a modular way to facilitate the easy addition of new error messages. Logging is also required for applications as well as batch/scheduled jobs.

2.1.17. Record Locking / Concurrent Users

2.1.17.1. Applications should be developed in such a way that users' changes do not clash with each other or create the potential for data loss/corruption.

2.1.18. System Testing

2.1.18.1. System testing shall consist of negative testing, as well as positive testing. During positive testing ("Testing to Pass"), the testers will ensure that a program behaves as it should (in terms of navigation, processing, reading and writing records, etc.). During negative testing ("Testing to Fail"), the testers will ensure that a program does not behave in a way that it shouldn't (e.g. allowing a past date to be entered into a future date field).

2.1.19. Regression Testing

2.1.19.1. Regression testing shall be done, where regression testing is any type of software testing that seeks to uncover new errors or regressions in existing functionality after changes have been made to the software, such as functional enhancements, patches or configuration changes. Regression testing ensures functionality that was working yesterday is still working today. New functionality should be added to a system without impairing existing functionality or introducing bugs.

2.1.20. Load Testing/Volume Testing

2.1.20.1. The load/volume testing that is performed on an application shall be reflective of the demands that could reasonably be expected to occur when the application goes into production. The testing should try to anticipate future system growth, data growth, and an increase in the number of active users.

2.1.21. Business Requirements – Traceability

2.1.21.1. All business requirements that have been captured and agreed upon by the project stakeholders shall be fully met in the final version of the application that is transitioned over to Application Services; and

2.1.21.2. All required system functionality shall also be fully satisfied by this final version.

2.1.22. Transition To Support Personnel

2.1.22.1. The necessary server environments to support an application (development, test/staging, and production) shall be fully constructed prior to transition;

2.1.22.2. The necessary server environment shall be entirely consistent with each other with respect to Operating Systems, software versions, database versions, environment hardening, configuration, etc; and

2.1.22.3. The Application Services resources supporting an application shall be granted access to development, test/staging, and production environments (as appropriate) prior to transition.

2.2. Checklists

Each section of the Application Quality Assurance Checklist template must be completed in full. If a particular section is not applicable to this project, then you must Check **Not Applicable (NA)** and provide a reason in the remark section. As well, if the answer is 'No' for any of the checklist items below, please explain why.

	Checklist Items	**Yes**	**No**	**NA**	**Remarks**
Development					
	1. Development Framework				
1.1	Has the application been developed with the most recent version of the framework for the chosen technology?				
	2. Development IDE				
2.1	Has the application been developed using an Integrated Development Environment as per the Integration Architecture?				
	3. Decoupling Business Logic from the Presentation Layer				
3.1	Is the presentation layer of the application free from business logic?				
3.2	Has the presentation layer of the application been developed in accordance with prevailing industry standards?				
	4. Certificates / Environment Software				
4.1	Has all proprietary and copyrighted software been properly licensed for government use?				
4.2	Have special software/certificate requirements been documented?				
4.3	Does the documentation provide expiration dates and instructions for renewal?				
4.4	Is the system / application free from trial versions of software?				
Database					

Checklist Items		Yes	No	NA	Remarks
5. Database Design					
5.1	Have the database tables been normalized?				
5.2	Have the database design aligned with institutional data dictionary?				
5.3	Keys based on sequence numbers have unique sequences?				
5.4	Are all keys and required fields set to 'not null' in all tables of the database?				
5.5	Have triggers, stored procedures, sequences and constraints been properly utilized?				
Codes					
6. Modularized Code with No Duplication					
6.1	Is the application modularized?				
6.2	Has code duplication been avoided?				
7. Consistency of Code					
7.1	Is the code written in a consistent manner throughout the application?				
7.2	Have all developers followed the same coding style and naming conventions?				
7.3	Have all developers followed the coding best practices as set out by the organization which owns the technology?				
8. Code Comments					
8.1	Does all application code include sufficient comments for support personnel?				
8.2	Does each code unit have its own brief and accurate description?				
8.3	Does all difficult/complex scenarios have comments?				
9. Hard-Coded Values					
9.1	Does the application code avoid use of hard-coded values?				
9.2	Do all hard-coded values reside exclusively within configuration and constant, centralized locations? (Central Locations that enable changes without recompiling source code)				
10. Source Code					

Checklist Items		Yes	No	NA	Remarks
10.1	Has the final approved version of the Application Code been provided to Application Services for use and maintenance during the Transition Period?				
10.2	Has Application Code developed and reviewed as per coding standards?				
10.3	Do reusable source codes of the application updated in Knowledge base repository?				
10.4	Are unit test cases prepared, reviewed and approved? Are the source codes being tested with unit test cases?				
10.5	Has a test build been completed by Application Services using the code that has been handed over?				
10.6	Has a copy of the version of Open-Source Code used by the application been provided to Application Services for retention? (Links are not recommended)				
10.7	Have the 3rd party developer code / plug-ins (e.g., Axis2, Eclipse) been identified and provided to Application Services for the continued maintenance of the application? (Links to the utility not satisfactory, 3rd party products need to be provided)				
Validations					
11. Field Validations					
11.1	Are fields being checked for the correct type (e.g., date, integer, etc.) and the correct range of values (e.g., 1 – 12 for month)?				
11.2	Are field values being validated with regular expressions where possible (e.g., validating email addresses and dates for valid formats)?				
11.3	Do the validations resulting in error messages prevent data from being written to persistent storage (databases, files, etc.)?				
11.4	Are the validations being performed within the business logic, as well as on the presentation layer?				
11.5	Have the validations been written so that users cannot bypass them?				

Checklist Items		Yes	No	NA	Remarks
11.6	Are all field lengths and types within the application consistent with the column lengths and types declared within the underlying database tables?				
11.7	Are user inputs being sanitized (without exceptions) according to e-Government Interoperability Framework (eGIF) recommendations?				
12. Dates					
12.1	Does the application validate dates in a way that is consistent with the system design specifications and business rules?				
12.2	Do all relevant dates include a timestamp?				
Authentication					
13. Passwords					
13.1	Does the system have functionality to allow the user to revise their password and force user account expiry?				
13.2	Does the system support protected storage of passwords with privileged user access? The system should not support passwords in clear text embedded either in the application code, automated scripts, or the database?				
13.3	Does the system meet the standard password requirements? Refer to the Security Architecture documents: Password Management				
13.4	Are the passwords in the production environment different than those in a non-production environment?				
13.5	Are all vendor supplied default passwords revised prior to placing the application in a production environment?				
Logging and Errors					
14. Logging and Auditing					
14.1	Based on the application's Information Security Classification, does the application meet the logging functional control requirements?				

Checklist Items		Yes	No	NA	Remarks
14.2	Based on the application’s Information Security Classification, does the application meet the auditing functional control requirements?				
15.	**Error Handling – End User**				
15.1	Does the application handle all the errors that could reasonably be expected to occur?				
15.2	Do the error messages contain minimal but meaningful information?				
15.3	Does the application avoid displaying system information in error messages?				
15.4	Are the error messages kept in a single location?				
16.	**Error Logging**				
16.1	Are all errors for the application being logged?				
16.2	Is logging being done on each server tier?				
16.3	Are the logs kept in a single location / directory / database?				
16.4	Are the logged errors specific enough to assist support personnel in troubleshooting production problems?				
16.5	Is the code that logs the error messages written in a modular way?				
16.6	Are the log files free of personally sensitive or identifiable information?				
17.	**Record Locking / Concurrent Users**				
17.1	Have precautions been taken to avoid data clashes?				
Testing					
18.	**System Testing**				
18.1	Did the application pass all positive tests?				
18.2	Did the application pass all negative tests?				
18.3	Have client testers completed the formal test plan in its entirety?				
18.4	Did the application pass all tests included in the formal test plan?				
18.5	Have all positive / negative test cases and test case results been documented?				
19.	**Regression Testing**				

Checklist Items		Yes	No	NA	Remarks
19.1	As new capability is introduced, is the new capability tested?				
19.2	Have all previous tests been reconducted with the results compared against expected results?				
19.3	Is every capability of the software supported with a test case and is the test case added to the test case library to support final and future system testing?				
19.4	As bugs are detected and fixed, is the test that exposed the bug recorded and regularly re-tested after subsequent changes are applied to the application?				
20.	**Load Testing/Volume Testing**				
20.1	Has the application been tested with a large number of concurrent users (i.e. a number of users that is representative of peak system usage)?				
20.2	Has the application been tested with large numbers of concurrent transactions (i.e. a number of transactions that is representative of peak system usage)?				
20.3	Did the system perform well with a large number of concurrent users?				
20.4	Did the system perform well with a large number of concurrent transactions?				
Business Requirements					
21.	**Business Requirements – Traceability**				
21.1	Have all of the business requirements been met by the finished application?				
21.2	Has all of the required functionalities been met by the finished application?				
21.3	Has each required requirement been mapped with other related requirements (dependencies), if exist?				
21.4	Has each requirement been realized in the system detailed design of the application?				
21.5	Does each requirement have prepared test case(s) such as Unit test case(s), System test case(s)?				
21.6	Does the finished application with all required requirements identified with proper release/ version naming?				

Checklist Items		Yes	No	NA	Remarks
System Support					
22.	**Transition To Support Personnel**				
22.1	Have accounts been created on all servers for the appropriate support personnel?				
22.2	Have the necessary firewall rules been added to allow Application Services support personnel to access the relevant servers (i.e., via the Jump Box)?				
22.3	Have all server environments (development, test/staging, and production) been fully created?				
22.4	Are all of the server environments entirely consistent with each other?				

3. IMPLEMENTATION, REVIEW AND ENFORCEMENT

This document shall:

3.1. Effective upon being reviewed by e-GA Management and signed by the Director General on its first page;

3.2. Subjected to review at least once every three years or whenever necessary changes are needed; and

3.3. Consistently complied with, any exceptions to its application must duly be authorized by the Director General.

4. GLOSSARY AND ACRONYMS

4.1. Glossary

None.

4.2. Acronyms

Abbreviation	**Explanation**
e-GA	e-Government Authority
OWASP	Open Web Application Security Project
SQA	Software Quality Assurance
SSL	Secure Socket Layer
XML	Extensible Mark Up Language

5. RELATED DOCUMENT

5.1. eGovernment Interoperability Framework - Standards and Technical Guidelines *(eGA/EXT/GIF/001)*

5.2. eGovernment Business Architecture - Standards and Technical Guidelines *(eGA/EXT/BSA/001)*

5.3. eGovernment Application Architecture - Standards and Technical Guidelines *(eGA/EXT/APA/001)*

5.4. eGovernment Information Architecture - Standards and Technical Guidelines *(eGA/EXT/IFA/001)*

5.5. eGovernment Infrastructure Architecture - Standards and Technical Guidelines *(eGA/EXT/IRA/001)*

5.6. eGovernment Security Architecture - Standards and Technical Guidelines *(eGA/EXT/ISA/001)*

6. DOCUMENT CONTROL

Version	Name	Comment	Date
Ver. 1.0	Government Software Applications Quality Assurance Guidelines and Checklist	Document Creation	March 2016
Ver. 2.0	Government Software Applications Quality Assurance Guidelines and Checklist	Aligning the document with e-Government Act No 10, 2019	May 2024