THE UNITED REPUBLIC OF TANZANIA

PRESIDENT'S OFFICE - PUBLIC SERVICE MANAGEMENT

e-GOVERNMENT AGENCY

Document Name

Government Software Applications Quality Assurance Checklist

Document Title

eGA/EXT/APA/002

APPROVAL	Name	Job Title/ Role	Signature	Date
Approved by	Dr. Jabiri Bakari	Chief Executive Officer		18/03/16

THE UNITED REPUBLIC OF TANZANIA
PRESIDENT'S OFFICE - PUBLIC SERVICE MANAGEMENT
e-GOVERNMENT AGENCY

1. OVERVIEW

1.1. Introduction

The e-Government Agency (eGA) is established under the Executive Agencies Act No.30, 1997, Cap. 245 as a semi-autonomous Institution under President's Office Public Service Management. eGA is charged with the mandate of providing coordination, oversight and provision of e-Government initiatives and enforcement of e-Government standards to Public Institutions. In executing its duties, eGA shall implement and maintain a coordinated government operations for Information and Communication Technology (ICT) that include the formulation of standards, technical guidelines and procedures to effectuate the purposes of the Agency.

The Application Quality Assurance Checklist is intended to ensure "Custom-Built" applications 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. Review and internal control checking refer to verification of the processes being implemented against the plans, standards, procedures, guidelines, commitments etc.

1.2. Purpose

The checklist assists to ensure effective and efficient quality assurance of processes in an application development life cycle through facilitation, review and internal control checking. This document forms part of *eGovernment Application Architecture - Standards and Technical Guidelines (eGA/EXT/APA/001).*

1.3. Scope

This document applies to 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 is also applies to the project ICT Project team members during the design stage to ensure the developed application meets the quality standards at the execute stage.

2. APPLICATION QUALITY ASSURANCE CHECKLIST ITEMS

2.1 Important Notice

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

2.2 The Checklist

Checklist Items		Yes	No	NA	Remarks
1. Development Framework Applications should be developed using the most recent e-Government Interoperability Framework (eGIF) and Application Architecture as stated in e-Government Related Standards and Guidelines ***(http://www.ega.go.tz/index.php/publications/index/4)*** Other framework versions pose compatibility, security, and upgrade risks for Application Services.					
1.1	Has the application been developed with the most recent version of the framework for the chosen technology?				
2. Development IDE Applications should 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	Has the application been developed using an Integrated Development Environment as per the Integration Architecture?				
3. Decoupling Business Logic from the Presentation Layer					

Checklist Items		Yes	No	NA	Remarks
Whenever possible, developers should 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). Also, the Presentation Layer of Web applications should be developed using prevailing industry standards (e.g. using Stylesheets to position and control presentation elements, using relative positioning instead of absolute positioning, etc.).					
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. Record Locking / Concurrent Users 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.					
4.1	Have precautions been taken to avoid data clashes?				
5. Passwords A password helps authenticate a user when accessing a software application. Adherence to appropriate password management will help maintain the confidentiality, integrity, availability of the data maintained by the software application and reduce the risk of inappropriate access and use.					
5.1	Does the system have functionality to allow the user to revise their password and force user account expiry?				
5.2	Does the system support protected storage of passwords with privileged user				

Checklist Items		Yes	No	NA	Remarks
	access? The system should not support passwords in clear text embedded either in the application code, automated scripts, or the database?				
5.3	Does the system meet the standard password requirements? Refer to the Security Architecture documents: Password Management				
5.4	Are the passwords in the production environment different than those in a non-production environment?				
5.5	Are all vendor supplied default passwords revised prior to placing the application in a production environment?				
5.6	Are passwords for privileged accounts different than passwords for non-privileged accounts?				
6. Logging and Auditing 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					
6.1	Based on the application's Information Security Classification, does the application meet the logging functional control requirements?				
6.2	Based on the application's Information Security Classification, does the application meet the auditing functional control requirements?				
7. Modularized Code With No Duplication As much as possible, code should be organized into small, separate modules to avoid code duplication and to make future code changes easier to implement.					
7.1	Is the application modularized?				

Checklist Items		Yes	No	NA	Remarks
7.2	Has code duplication been avoided?				
8. Consistency of Code Code sections with similar functionality should 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.					
8.1	Is the code written in a consistent manner throughout the application?				
8.2	Have all developers followed the same coding style and naming conventions?				
8.3	Have all developers followed the coding best practices as set out by the organization which owns the technology?				
9. Code Comments Code sections should 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. Any potentially confusing / non-intuitive sections of code should be commented thoroughly.					
9.1	Does all application code include sufficient comments for support personnel?				
9.2	Does each code unit have its own brief and accurate description?				
10. Error Handling – End User Error messages presented to the end user should contain only that information which will allow the user to take corrective action (e.g. "Invalid date, please reenter in YYYY-MM-DD format"). In the case of unhandled exceptions, messages should 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					

Checklist Items		Yes	No	NA	Remarks
information. Avoid messages that could potentially lead to system exploitation (e.g. "Incorrect Login" is acceptable while the message "Incorrect Password" is not). Error handling logic should be robust enough to gracefully and meaningfully handle all errors which could be reasonably expected to occur from user interactions with the system. The text for error messages should be contained in a single location within the application code or database to facilitate quick additions and modifications by support staff.					
10.1	Does the application handle all the errors that could reasonably be expected to occur?				
10.2	Do the error messages contain minimal but meaningful information?				
10.3	Does the application avoid displaying system information in error messages?				
10.4	Are the error messages kept in a single location?				
11. Error Logging Application errors should 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. Log files for individual server tiers (i.e. Web and Application tiers) should be kept in a single directory on each server. Also, log files should be saved on a daily basis with a time-date stamp on each file. The error messages that are logged should contain information that is useful for support personnel (absolutely no sensitive or personal data), such as which module of code encountered the error and what the specific error was. Meaningful and detailed error					

Checklist Items		Yes	No	NA	Remarks
messages are particularly important when troubleshooting unknown/unexpected errors. These should definitely be captured and logged. Logging is also required for applications as well as batch/scheduled jobs. Logging logic within applications should be written in a modular way to facilitate the easy addition of new error messages.					
11.1	Are all errors for the application being logged?				
11.2	Is logging being done on each server tier?				
11.3	Are the logs kept in a single location / directory / database?				
11.4	Are the logged errors specific enough to assist support personnel in troubleshooting production problems?				
11.5	Is the code that logs the error messages written in a modular way?				
11.6	Are the log files free of personally sensitive or identifiable information?				
12. Field Validations Where possible, validations should 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 should also be done within Java classes on the business layer. Also, validations should 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					

Checklist Items		Yes	No	NA	Remarks
Framework (eGIF) and Information Architecture					
12.1	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)?				
12.2	Are field values being validated with regular expressions where possible (e.g. validating email addresses and dates for valid formats)?				
12.3	Do the validations resulting in error messages prevent data from being written to persistent storage (databases, files, etc.)?				
12.4	Are the validations being performed within the business logic, as well as on the presentation layer?				
12.5	Have the validations been written so that users cannot bypass them?				
12.6	Are all of the field lengths and types within the application consistent with the column lengths and types declared within the underlying database tables?				
12.7	Are user inputs being sanitized (without exceptions) according to e-Government Interoperability Framework (eGIF) recommendations?				
13. Dates When testing functionality that is built around date checks, the testers should use date values that occur in the past, on the target date, and in the future. Dates should also 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?). When dates are recorded in a database or log, they should include a timestamp 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.					
13.1	Does the application validate dates in a way that is consistent with the system				

Checklist Items		Yes	No	NA	Remarks
	design specifications and business rules?				
13.2	Do all relevant dates include a timestamp?				
14. Hard-Coded Values Hard-coding of server names, database names, domain names, IP addresses, etc. within application code should 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. Fixed values that are repeatedly used throughout application code should 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.					
14.1	Does the application code avoid use of hard-coded values?				
14.2	Do all hard-coded values reside exclusively within configuration and constant, centralized locations? (Central Locations that enable changes without recompiling source code)				
15. System Testing System testing should 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).					

Checklist Items		Yes	No	NA	Remarks
15.1	Did the application pass all positive tests?				
15.2	Did the application pass all negative tests?				
15.3	Have client testers completed the formal test plan in its entirety?				
15.4	Did the application pass all tests included in the formal test plan?				
15.5	Have all positive / negative test cases and test case results been documented?				
16. Regression Testing 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.					
16.1	As new capability is introduced, is the new capability tested?				
16.2	Have all previous tests been recon ducted with the results compared against expected results?				
16.3	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?				
16.4	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?				
17. Load Testing/Volume Testing The load/volume testing that is performed on an application should 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					

Checklist Items		Yes	No	NA	Remarks
increase in the number of active users.					
17.1	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)?				
17.2	Has the application been tested with large numbers of concurrent transactions (i.e. a number of transactions that is representative of peak system usage)?				
17.3	Did the system perform well with a large number of concurrent users?				
17.4	Did the system perform well with a large number of concurrent transactions?				
17.5	Are end-users satisfied with the application's performance and responsiveness during everyday use?				
18. Certificates / Environment Software 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.) should be documented in the Operations Procedure Manual. Documentation should include the relevant expiration dates and the processes that must be followed for renewal. Also, application deployments in production environments should not be comprised of any trial versions of software. All proprietary and copyrighted software should be properly licensed for Government use					
18.1	Has all proprietary and copyrighted software been properly licensed for government use?				
18.2	Have special software/certificate requirements been documented?				
18.3	Does the documentation provide expiration dates and instructions for renewal?				
18.4	Is the system / application free from trial versions of software?				
19. Business Requirements – Traceability					

Checklist Items		Yes	No	NA	Remarks
All of the business requirements that have been captured and agreed upon by the project stakeholders should be fully met in the final version of the application that is transitioned over to Application Services. All required system functionality should also be fully satisfied by this final version.					
19.1	Have all of the business requirements been met by the finished application?				
19.2	Has all of the required functionality been met by the finished application?				
19.3	Has each required requirement been mapped with other related requirements (dependencies), if exist?				
19.4	Has each requirement been realized in the system detailed design of the application?				
19.5	Does each requirement have prepared test case(s) such as Unit test case(s), System test case(s)?				
19.6	Does the finished application with all required requirements identified with proper release/ version naming?				
	20. Source Code				
20.1	Has the final approved version of the Application Code been provided to Application Services for use and maintenance during the Transition Period?				
20.2	Has Application Code developed and reviewed as per coding standards?				
20.3	Do reusable source codes of the application updated in Knowledge base repository?				
20.4	Are unit test cases prepared, reviewed and approved? Are the source codes being tested with unit test cases?				
20.5	Has a test build been completed by Application Services using the code that has been handed over?				
20.6	Has a copy of the version of Open Source Code used by the application been				

Checklist Items		Yes	No	NA	Remarks
	provided to Application Services for retention? (Links are not recommended)				
20.7	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)				
21. Database Design Industry best practices should be followed in the design of databases for production applications: tables normalized, exceptions documented, constraints enforced, and required fields completed (nulls not permitted). Also, if table keys are based on sequence numbers, each table should have its own sequence.					
21.1	Have the database tables been normalized?				
21.2	Keys based on sequence numbers have unique sequences.				
21.3	Are all keys and required fields set to 'not null' in all tables of the database?				
21.4	Have triggers, stored procedures, sequences, and constraints been properly utilized?				
22. Transition To Support Personnel The necessary server environments to support an application (development, test/staging, and production) should be fully constructed prior to transition and should be entirely consistent with each other with respect to Operating Systems, software versions, database versions, environment hardening, configuration, etc. The Application Services resources supporting an application should be granted access to development, test/staging, and production environments (as appropriate) prior to transition.					
22.1	Have accounts been created on all servers for the appropriate support personnel?				

Checklist Items		Yes	No	NA	Remarks
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?				
23. Transition to Support Personnel The necessary server environments to support an application (development, test/staging, and production) should be fully constructed prior to transition and should be entirely consistent with each other with respect to Operating Systems, software versions, database versions, environment hardening, configurations etc. The Application Services resources supporting an application should be granted access to development, test/staging, and production environments (as appropriate) prior to transition.					
23.1	Are all of the server environments entirely consistent with each other?				

3. STANDARDS IMPLEMENTATION, REVIEW AND ENFORCEMENT

3.1. This document takes effect once signed and approved in its first page.

3.2. This document is subject to review at least once every three years.

3.3. This Documents need to be complied to as directed in "*Mwongozo wa Matumizi Bora, Sahihi na Salama ya TEHAMA Serikalini, 2016*".

4. ACRONYMS

Abbreviation	**Explanation**
eGA	e-Government Agency
OWASP	Open Web Application Security Project
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	eGA	Document for Approval	March 2016