REQUIREMENTS

REQUIREMENTS

1. INTRO

1.1. INTRO

1.2. Audience

This document is aimed for any potential and actual developers of the qto application. Product Owners, Developers and Architects working on the application MUST read and understand this document at least to the extend of their own contribution for the application.

1.3. Related documentation/

This document is part of the QTO application documentation-set, which contains the following documents:
 - ReadMe - the initial landing readme doc for the project
 - UserStories - the collection of user-stories used to describe "what is desired"
 - End-User Guide - the guide for the usage of the UI (mainly) for the end-users
 - Requirements - the structured collection of the requirements
 - SystemGuide - architecture and System description
 - DevOps Guide - a guide for the developers and devops operators
 - Installation Guide - a guide for installation of the application
 - Concepts - the concepts doc

You can access all the latest qto documentation from qto site:
https://qto.fi
in it's native format.
All the documents are updated and redistributed in combination of the current version of the application in both md and pdf file format and can be found under the following directories:
 - doc/md
 - doc/pdf

2. DEPLOYABILITY

The qto must be easily deployable on any Unix like OS.
Windows family based OS'es are explicitly out of the scope of the qto tool.
Any qto instance should be configurable as easily as possible for its version.

2.1. DevOps deployability no longer than a week release cycle

The qto system should provide the CI infrastructure for the capability to perform frequent releases, with release cycles no longer than a week. For larger releases 2 weeks cycle are acceptable too, with no more than 3 larger releases in a row.

2.2. Automated AWS deployment in less than an hour

The qto system should be automatically deployable to aws in less than an hour, so that the deployment operator should execute no more than 5 documented commands.

2.3. Full deployment in less than an hour

The full System should be ready for use by end-users in the latest Ubuntu LTE OS in less than an hour. The whole deployment process MUST BE as automated as possible.

2.3.1. A working instance deployment by simple unzip command

The qto tool could be deployed by a simply unzip of the full package into a host having the proper binary configuration, which must have all of the documentation and scripts to provide assistance for the setup and the configuration of the tool as well as the initial data to populate the qto database.

2.3.2. Binary prerequisites check script

All the binaries which are required for the running of the tool must be checked by a user-friendly binaries prerequisites check script.

2.3.3. Required Perl modules installation

All the required Perl modules must be part of the deployer script. The Perl modules should be installed as non-root user.

2.3.4. Installation documentation

The installation of the required Postgres db must be documented in the DevOps guide, which should have the markdown version in the doc directory of the deployment package.

2.4. Full application clone creation in 5min

A DevOps operator should be able to perform an application clone (having app-name changed to new-app-name etc.) of the Qto application in less than 5 minutes.

2.4.1. Single shell call for postgres db creation and initial data load

The creation of the Postgres database of a qto project should be doable via a single shell call.

2.4.2. One liner for single restore for both full db and inserts only

The full database should be loadable form a db dump either from a full dump or from the db-inserts dump only.

2.5. Singleton configuration

Whenever there is a configuration entry indicating part of the configuration of an application instance it should be stored in 1 and only 1 place in the configuration file of the instance, so that any person not familiar with the internal logic of the application could find all the configuration entries in one and only one place including the secrets.

3. RELIABILITY AND STABILITY

3.1. Zero tollerance towards crashing

Crashing in normally configured and operating environment must not be tolerated, as soon as any crash has occurred a bug must be registered and the bug set with the highest possible prio towards the features pipeline.

3.2. Zero tolerance towards bugs

All bugs and inconsistencies must be dealt with top priority by passing new features implementation. Should the average amount of bugs increase after a release a purely bug fixing release should follow. Any reasonable refactoring could and should bypass the new features implementation. tolerance

3.3. e2e verticality

The qto source code artefact must contain both the documentation and the references for the published and running qto application AND data, so that any potential developer or user of the application could easily setup a new instance or simply use one relying on the fact that the full stack has been tested and applied according to that very latest release he or she is using.

3.4. Daily backups

The daily backups of all instance project databases data and secrets should be performed automatically as indispensable part of the functioning of the application.

3.5. Logging

The application should support configurable logging to STDOUT and STDERR for the following levels - debug, info, warn, trace.

3.6. Full backup to the cloud in less than 5 minutes

A full backup of the software, configuration and data for the qto and/or another project database should be doable in less than 5 minutes. The backup should be easily searchable from the cloud as well.

4. USABILITY

The interaction with each endpoint and interface of an application instance should be as user-friendly as possible.
As abstract as it may sound the tool must be multi-dimensionally and vertically integrated regarding the questions what, how and why towards a new person interacting with the tool by the usage of code comments , links from the documentations and uuids to be used for simple greping from the docs till the source code.

4.1. Application wide top-bar usability

The top-bar of the application must be ALWAYS selected during a page load and particularly the omni-search box, so that the user could straight way start quick filtering the textual content of ANY page by typing at least 3 chars in the search box.
The top-bar must be visible on ALL pages, except those aimed for printing.

4.2. Application wide left menu usability

The left menu must be ACCESSIBLE from the SAME button on EVERY non-printable page of the application from the top bar. The left-menu must contain the most commonly used links in the application.

4.2.1. Left menu global configurability

The Admin of an instance must be able to fully configure the content of the left menu.

4.2.2. Left menu personal customisability

Any user should be able to personally configure which are the left menu links, he/she would like to use. The personal links should be capable of fully overriding the links set by the administrator.

4.3. UI usability

The interaction of with the application UI must be as effortlessly, quickly and user-friendly as possible.

4.3.1. Application wide top-bar usability

The top-bar of the application must be ALWAYS selected during a page load and particularly the omni-search box, so that the user could straight way start quick filtering the textual content of ANY page by typing at least 3 chars in the search box.
The top-bar must be visible on ALL pages, except those aimed for printing.

4.3.2. Login page usability

The login page must contain clearly on which instance a user is logging in. The login must be executable both with enter and click of the GO button. The login page must load in less than 0.3s. The login after that to the home / landing page must not take more than 0.5 seconds. The error msgs must be as concise, but clear and explanatory as possible.

4.3.3. Landing / home page usability

The landing / home page must clearly indicate that the user has logged in. It must contain clear UI element(s) to indicate where to go from here. The landing page might contain some additional informative content.

4.3.4. View page usability

The view page must load in less than 0.3 s.

4.4. Oneliner shell calls

The interaction of the application on the shell should be designed and implemented so that most of the features and bigger entry points should be accessible via one-liners on the shell - for example the testers should be able to lunch all the unit-tests via a single one line call. The integration tests should be triggerable via single oneline call.

4.4.1. Database recreation and DDL scripts run one-liners

The developers should be able to create the database via a single oneline call.

4.4.2. Table(s) load via aa single one-liner

The developers should be able to load a table to the database via a single oneline call.

5. SCALABILITY

5.1. Feature scalability

The addition of new features should be as scalable as possible. The UI and control flow should be as generic as possible.

5.2. Setup scalability

The creation of new instances of the application should be as easy and automated as possible.

5.3. Projects databases scalability

Each instance of the qto application must be able to connect to one or many project databases which DDL schemas matching the current api of the application.

5.4. Qto application clones scalability

The qto application must support cloning - that is "forking" into new applications with different names.

5.5. Db model scalability

The list page must support ANY db tables DDL format complying to the guid PK and id uniq key interface. A sysadmin should be able to alter a table, reload the meta_tables or the meta_columns tables from the ui to the reload the meta-data for the users to be able to use the updated tables without disruptions or application layer restarts.

6. PERFORMANCE

6.1. Page load maximum time

Each page of the application containing less than 2000 textual rows without pictures MUST load for less than 0.3 seconds.
Any new feature which does not meet this requirement should be disregarded or implemented into a clone of the application with different name (see the cloning / forking section bellow).

6.2. Login, logout

Every login and logout operation MUST complete in less than 0.3 seconds in modern network environments.

7. MULTI-INSTANCE OPERABILITY AND DEPLOYABILITY

7.1. Environment type self-awareness

Each deployed and running instance of the qto must "know" its own environment type - dev, tst, qas or prd to comply with the multi-instance architecture on a single host.

7.2. Cross running between instances of different types

The application layers should support as much as possible cross running between different application layer instances and database instances - for example a dev application layer should be able to fetch data from a prd database.

8. UI REQUIREMENTS

The UI of the application must be fast, responsive, easy and pleasant to use.

8.1. CRUDs

The System must provide the needed UI interfaces to Create , Update , Delete and Search items in the system for the users having the privileges for those actions
Any modelled item in the database must be capable for:
- create
- update
- delete
- search

8.1.1. Execution time

The full execution time of any crud operation (create, update, delete, search) from the end-user of the UI point of view should be less than 0.3 seconds

8.1.2. Visual indication

The System should not show the so called ok messages, but only error messages (with added verbosity on the console level), yet the UI should be as responsive that the end-user would easily understand when an item has been created, updated or deleted.

8.2. Clarity on errors

The UI must present every error in a clear and concise way, so that the end-user would understand that an error has occurred, however no msgs should be displayed when the data is saved properly.

8.3. Information search

The UI must enable easy to use and effective search for the end-users from all but the login page.

8.3.1. Global project search

The UI must support global per project (i.e. single project database) text search from all but the login page.

8.3.2. Quick search per page

Each information on each different type of search must be filterable from the omni search box - both the page content and the auxiliary left and/or right menu.

8.4. Login page requirements

All login error msgs should be clear and displayed with red colour.

8.5. List page requirements

Each information on list page must be filterable from the omni search box - both the page content and the auxiliary left and/or right menu.

8.5.1. View page requirements

Each information on view doc page must be filterable from the omni search box - both the page content and the auxiliary left and/or right menu.

9. INTEROPERABILITY

The qto application should support export and import from different data formats from both client and server side.

9.1. Data import

9.1.1. Import from xls

The qto application should support a single xls data import.

9.1.2. Import from json format

The qto application should support a single shell call json file to table import

9.2. Data export

The qto application must support export to different data formats via both the web interface or terminal access.

9.2.1. Export to pdf

Any user having tcp access to an up-and-running qto instance must be able to export all or single item doc from that or another qto instance with tcp / ip connectivity into pdf files into a pre-configurable docs root directory both via ui or from an automation script.

9.2.2. Export to md

Any user having tcp access to an up-and-running qto instance must be able to export all or single item doc from that or another qto instance with tcp / ip connectivity into md files into a pre-configurable docs root directory both via ui or from an automation script.

9.2.3. Export to xls

Any user having tcp access to an up-and-running qto instance must be able to export all or single item doc from that or another qto instance with tcp / ip connectivity into Microsoft xls files into a pre-configurable docs root directory both via ui or from an automation script.

9.2.4. Export to Microsoft docx

Any user having an up-and-running qto instance must be able to export all or single item doc from that or another qto instance with tcp / ip connectivity into Microsoft docx files into a pre-configurable docs root directory both via ui or from an automation script.

9.2.5. Export to json

Any user having an up-and-running qto instance must be able to export all or single table from that or another qto instance with tcp/ip connectivity into json files.

10. SECURITY

A well operated instance of the qto application should have security corresponding to the data sensitivity it is operating on.
For complete walkthrough of instance security check the following document: security_checklist_doc-0

10.1. Authentication

There should be the following 2 modes of authentication:

10.1.1. Non-authentication mode

Any qto instance should support a non-authentication mode - that is all users having http and/or https access could perform all the actions on the UI without any restrictions, that is the customer organisation wanting own custom solution for authentication and authorisation should be able to run an instance with non-authorisation mode.

10.1.2. Simple Native authentication mode

All registered users should have access to all but users related data.

If a user is not registered the error msg to the login should prompt him which e-mail to contact to be registered (which will be the e-mail of the product owner instance).

If the admin user is able to impersonate another user it must simply mean that he/she has done that on purpose (aka maliciously)
The sessions of different dev, tst and prod app layer instances should not intermix within the multiple open processes / threads of the same browser of the same user.

10.1.2.1. User email and password matching for login success

Users should login with email and password. Users' names and other personal data MUST NOT be tracked by the application.
Unregistered users should have access to the login page only.

10.1.2.2. Blowfish encryption for the passwords

The application must match the passwords via blowfish encryption and store the authentication details into the session of default of 10h.

10.1.2.2.1. Passwords sensibility

The regular users should see only their credentials. Only the admin user should see all the users credentials , but with the passwords encrypted.

10.1.3. JSON web token authentication

The qto application should support native web tokens based authentication, by using as login a valid user e-mail and password, stored in the qto instance database.

The qto should support SSO authentication as described in the following RFC's.
The Users should be authenticated by means of the most simplest OAauth2.0 authentication flow:
https://tools.ietf.org/html/rfc6749#section-5.1

10.1.3.1. Login

The login must be done against email and password over https. Should the email and the pass not match, a 401 http code (UnAuthorised) must be returned.

10.1.3.2. Must be stateless and thus horizontally scalable

The JSON web token authentication must be stateless to enable both process (aka run-time) and hosts scalability. The JWT must be signed with private key against tampering.
Token verification must be done without db lookup and only in-memory.
Neither sticky sessions should be used ...

10.1.3.3. Must not use permanent cookies

Because permanent cookies might be hacked by js scripts which are running in the same browser instance ...

Must use HttpOnly cookies:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

10.1.3.4. Must not use localStore, but Authorisation header

But rather some combination of httpOnly session and JWT , same-origin
Than CORS will not be an issue - https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS , must stay operational even if.

10.1.3.5. Must have arymetric signature

The JSON web tokens must be viewable from the other parties by . Must support public and private key decoding

10.1.3.6. The tokens must be self-contained

The tokens must be self-contained, i.e. the token must carry all the user data needed for the authorisation in is payload, but MUST not expose the site to XSS.
The tokens must be set in the Authorisation header.

10.1.3.7. Tokens expiration

The tokens could request refresh tokens once the expiration time is close.

10.1.3.8. Must support whitelisting

Whitelisting is the practice of explicitly allowing some identified entities access to a particular privilege, service, mobility, access or recognition.
A qto admin must be able to revoke access to user token or by username or by token id.

10.1.3.9. Must support black-listing

Blacklisting is the action of a group or authority, compiling a blacklist (or black list) of people, countries or other entities to be avoided or distrusted as not being acceptable to those making the list.[1] A blacklist can list people to be discriminated against, refused employment, or censored.
A qto admin must be able to revoke access from specific users and specific user tokens.

10.2. Authorisation

The Qto application should have authorisation as described in the RFC 6749.

10.3. Role-based Access control

The Application must restrict the system access ONLY to authorized users.

10.3.1. Traditional Unix permissions model per project database

The application must provide NON-ENFORCEABLE traditional permissions model for tables and/or table rows. The "non-enforceable" means that no overhead work must be enforced on organisations not-using the model at all - that is the default permissions will be 775 - meaning that the authenticated users must be able to both list(execute) and write to all the tables and rows in the project database(s), however so that the rest of the non-registered users must be able to list and read all tables and their rows.

10.3.1.1. Traditional Unix permissions model to tables

The application must provide the means for project owners to define custom read, write, execute (list) permissions on tables to per project database for table objects.

10.3.1.2. Traditional Unix permissions model to table rows

The application must provide the means for project owners to define custom read, write, execute (list) permissions on tables to per project database for table rows.

11. DOCUMENTATION

11.1. Documentation completeness

Each running instance MUST have the following documentation set :
- End User Guide
- Installation and Configuration Guide doc
- DevOps Guide doc
- Requirements doc
- System Guide doc
- UserStories doc
- Maintenance and Operations Guide doc

in the following formats:
- native qto view-doc page format (as soon as the instance is up-and-running)
- md format - as soon as the source code is downloaded

11.2. Documentation and code base synchronization

Each running instance MUST have its required documentation set up-to-date for it's release version. No undocumented or hidden features are allowed. Should any be missing or misreported a new issue must be created to correct those with top priority.

11.2.1. Requirements push

Whenever a project database meta-data is updated a new "do reload the current page" should be pushed on all the clients having currently session in the application …

/