REQUIREMENTS

## 1. INTRO

### 1.1. 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. 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.

## 3. 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.

#### 3.1. 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.

#### 3.2. Database recreation and DDL scripts run one-liners

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

#### 3.3. Testing one-liner call

The testers and the developers should be able to trigger all the unit or integration tests via a single one-line call.

## 4. RELIABILITY AND STABILITY

### 4.1. QTO clones scalability

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

## 5. PERFORMANCE

## 6. MULTI-INSTANCE OPERABILITY AND DEPLOYABILITY

## 7. UI REQUIREMENTS

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

### 7.1. 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

### 7.2. 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.

#### 7.2.1. Global project search

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

#### 7.2.2. 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.

#### 7.2.3. Login page requirements

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

#### 7.2.4. List page requirements

#### 7.2.5. View page requirements

## 8. SECURITY

A well operated instance of the qto application should have security corresponding to the data sensitivity it is operating on.

### 8.1. Authentication

There should be the following 4 modes of authentication:

#### 8.1.1. 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.

#### 8.1.2. 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.

##### 8.1.2.1. 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.

##### 8.1.2.2. 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.

#### 8.1.3. 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

### 8.2. Authorisation

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

### 8.3. Role-based Access control

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

#### 8.3.1. 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.

##### 8.3.1.1. 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.

##### 8.3.1.2. 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.

## 9. DOCUMENTATION

### 9.1. 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

### 9.2. 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.

#### 9.2.1. 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 …

## 10. WORKING PRINCIPLES

### 10.1. Personal responsibility

The whole design of the application as well as each system containing a running instance of it must support the principle for "personal responsibility" - aka for each error and / or faulty behaviour a concrete person must be available to whom the issue will be assigned.