FEATURES AND FUNCTIONALITIES

1. INTRO

1.1. Purpose

The purpose of this document is to present the features and functionalities set to the qto application for this current version.

1.2. Audience

This document could be of interest for any potential and actual users of the application. Developers and Architects working on the application MUST read and understand this document at least to the extend of their own contribution for it.

1.3. Related documentation

This document is part of the qto application documentation-set, which contains the following documents:
- readme - the initial readme file of the project
- enduser_guide_doc - the end users guide - mostly ui usage and concepts
- concepts_doc - contains the concepts of the application
- userstories_doc - the collection of userstories used to describe "what is desired"
- system_guide_doc - architecture and System description
- devops_guide_doc - a guide for the developers and devops operators
- installation_guide_doc - a guide for installation of the application
- features_doc - description of the current features and functionalities
- requirements_doc - description of the Requirements for the application

All the documents are updated and redistributed in combination of the current version of the application and could be found under the following directories:
doc/md
doc/pdf
according to the file format used for the documentation storage.

2. DEPLOYABILITY

The qto is 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 the version it has.

2.1. Full deployment in less than an hour

The full System is ready for use in a "blank" Unix-like OS host in less than an hour.
The installations instructions are done for Ubuntu 18.04 LTS, yet should you feel confortable with other Linux distros or even BSD Unix you should be able to complete it in less than 2 hours.

2.2. Easy docker based full deployment (deprecating)

This feature is being deprecated ... yet you could quickly re-implement it by changing the Dockerfiles versions .. By following the installation instructions in the installations_doc you can deploy on any docker running Unix-like OS the qto application running on a docker and Ubuntu 18.04 LTS with initially loaded database and data.

2.2.1. New version deployment by simple unzip

The qto tool could be deployed by a simply unzip of the full package, which must have all of the documentation and scripts to provide assistance for the setup and the configuration of the tool.

2.2.2. Oneliner for prerequisite binaries check

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

2.2.3. Installation documentation

The installation of the required mysql and postgres db must be documented in the DevOps guide, which should have both markdown and pdf versions in the doc directory of the deployment package.

2.3. A full application clone in less than 5 minutes

A DevOps operator is able to perform an application clone of the qto application in less than 5 minutes.

2.3.1. Shell script for postgres db creation

The creation of the postgres database is doable via a single shell call.

2.3.2. One liner for single restore and / or load

A qto db clone can be loaded via a single oneliner.

3. USER-FRIENDLINESS

The interaction with each endpoint and interface of an application instance is implemented to be as user-friendly as possible.
As abstract as it may sound the tool is 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 grepping from the docs till the source code.

3.1. 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.

3.1.1. Database recreation and DDL scripts run one-liners

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

3.2. Tables load via a single one-liner

The developers should be able to load one or many tables to the database via a single oneline call.

3.2.1. Testing one-liner call

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

3.2.2. Test messages user

Each test obeys the following convention:
- short message as descriptive within the context as possible - what is being tested
- a short technical example of the generated entry being tested (for example a dynamic url)
- a uuid to search for from the Feature document what exactly is being tested within the context of the features description.

4. STABILITY AND RELIABILITY

4.1. Absence of application crashing

Due to the stable architecture based on the based web framework out there - Mojolicious, the qto crashes experienced during the last years of operation have been extremely rare.
Qto is vertically integrated - once you have installed the full stack according to the latest released version the probability of experiencing an application crash are less than 0,01% of the time.

4.2. Daily backups

Daily backups are taken after the first shell action, run for the day, the daily backups oneliner could be scheduled via crontab as well.

4.3. Logging

The application supports fully configurable audit logging to both console (STDOUT, STDERR) and file.

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

A full backup for the data for the qto and/or another project database is doable in less than 5 minutes.

4.5. Stability based on actual running in the cloud since 2019-01-01

The main qto application instance has been up-and-running since the beginning of 2019 with receiving new versions in an average of 2 weeks per sprint.

5. SCALABILITY

5.1. Feature scalability

The addition of new features is scalable, as almost all of the components have been implemented according to the SOLID principle.

5.2. Multi-instance scalability

You can operated multiple versions of the qto on the same host, which will not interrupt each other because of the configurability of the application.

5.3. Documentation generation and data export

Single shell actions exist for :
- configurable pdf documentation generation
- configurable mdf documentation generation
- configurable microsoft docx documentation generation
- configurable xls tables export

5.4. Projects databases scalability

Each instance of the qto application can connect via tcp to multiple postgres databases running on the same db host configured in the instance configuration file.

6. PERFORMANCE

6.1. Page load times

Each page of the application containing less than 2000 loads for less than 0.5 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). The qto has been operated on quite modest hardware (check the second cheapest amazon ec2 instances for reference), yet the page load times vary from 0.3 till 0.6 seconds for the smaller pages and up till 1.5 seconds for the pages having more than 2000 items ...
Although the qto has not been explicitly designed for mobile devices the page load times on higher end phones on 4G networks are comparable with the upper 20% of the pages in the web, while running both the application layer and the db on a single aws instance based in Ireland ...

6.2. Login, logout

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

6.3. Ajax calls to back-end

Each back-end update from the UI takes no longer than 0.2 s. in a non-stressed qto instance, thus the look and feel of an qto instance is more like a desktop app and less like web app.

7. DEPLOYABILITY

7.1. New hosts binary provisioning

You can spawn new instances in the cloud from a client having the src code and needed terraform binary

7.2. Oneliner for version change on provisioned host

You could create a new instance of the qto having different version (which becomes automatically a dev environment) by issuing the following command:

```
bash src/bash/qto/qto.sh -a to-ver=0.7.7
```

7.3. Oneliner for environment change

You could change the environment type of your current instance by issuing the following command:

```
bash src/bash/qto/qto.sh -a to-env=tst
```

8. WEB USER INTERFACE FEATURES AND FUNCTIONALITIES

8.1. Single code base for all device formats

Although qto has not been explicitly designed for mobile phones and / or tablets it renders well on both high end mobile phones, and tablets over 4G networks.

8.2. List page features and functionalities

The list page is simply a slice of the data from ANY postgres table filtered on any criteria defined in the url of the browser.

8.2.1. List page performance

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

8.2.2. Navigating the list page

After the load of the list page the user can quickly cycle trough all the element of the page with the tab key on the keyboard. Focus on the search

8.2.3. Managing items - full CRUD

The System provides the needed UI interfaces to Create , Update , Delete and Search items in the database.

8.2.4. Quick search and filtering items

The user can quickly filter the items from the presented listing by typing in the omnisearch box ... The System will shrink the table so that only the rows having the string in the search omnibox will be presented. Once the string is deleted from the search omnibox the table data is restored to its original state.

8.2.5. Visual indication

The Systems does not present any ok messages for the operation of the list page, only errors are presented clearly on the top of the page (for example when one tries to update a string value into a cell with column accepting only integer values ...)

8.2.6. Print to pdf

You can print any of the queries from the list page by adding / changing the as url parameter from as=grid to as=print-table. Use the browser print to pdf feature to save the listing page into a pdf file.

8.2.7. Items interlinking

The users can link to any items by simply typing <<item>>-<<id>> in the description. For example requirements_doc-4

8.3. View page features

8.3.1. Managing items (beta)

The Qto application provides the needed UI interfaces to Create , Update , Delete items in the view documents UI for the users having the privileges for those actions.

8.3.1.1. Add an item in the doc view page UI (beta)

Users with the write privileges for the document can add an item in the doc view page just by right clicking on the title and selecting one of the 3 options:
- add sibling node - add an item which is on the same level in the hierarchy
- add parent node - add an item which is on 1 level up in the hierarchy
- add child node - add and item which is on 1 level bellow in the hierarchy

The new item appears straight after the origin title it was requested from.

8.3.1.2. Update item (beta)

You can:
- update item title content
- update item description
- update item src code if visible (you can make it visible by adding any non space content to it in the list page by right clicking the item number and choosing open in list)

8.3.1.3. Delete item (beta)

You can right click on an item and choose the remove node from the right click men.

8.3.1.4. Print to pdf

You can print any view doc by choosing right click view as pdf and choosing print to pdf file from the browser. Check the generate-pdf-docs shell action to automate this for each document configured in the export_files table.

8.3.2. Print to pdf

You can print any view doc by choosing right click view as pdf and choosing print to pdf file from the browser. Check the generate-pdf-docs shell action to automate this for each document configured in the export_files table.

9. SECURITY

9.1. Authentication

The qto application supports the following 2 modes of security:
- non authentication mode
- simple native authentication mode

9.1.1. Non-athentication mode

Any qto instance supports a non-authentication mode - that is all users having http access could perform all the actions on the UI without restrictions

9.1.2. Simple native authentication mode

A qto instance running under the simple native authentication mode stores the user credentials in the instance db. The passwords are encrypted via the using the Blowfish-based Unix crypt() hash function, known as "bcrypt" encrypting mechanism.

9.2. Authorisation

Only the SysAdmin of the System can add basic authentication and simple native mode users, thus regular users can see only their own credentials.

10. DOCUMENTATION

10.1. Documentation set

Each running instance has the following documentation set :
- ReadMe
- Features and Functionalities doc
- End User Guide
- DevOps Guide
- SystemGuide
- UserStories document
- Installation and Configuration Guide

In both "native qto format" and md file format in the doc/md directory of the project.

10.2. Documentation formats

The qto documentation is available in both md and pdf formats.

10.3. Documentation and code base synchronization

Each running instance has it's required documentation set up-to-date. 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.