Table of Contents

Table of Contents 1
REQUIREMENTS 2
1. INTRO 2
1.1. Purpose 2
1.2. Audience 2
1.3. Related documentation 2
2. DEPLOYABILITY 2
2.1. Full deployment in less than an hour 2
2.1.1. Deploy by simple unzip 2
2.1.2. Oneliner for prerequisite binaries check 2
2.1.3. Oneliner for Perl modules check 2
2.1.4. Installation documentation 2
2.2. A full application clone should be ready for less than 5 minutes 2
2.2.1. Shell script for postgres db creation 2
2.2.2. One liner for single restore 2
3. USER-FRIENDLINESS 3
3.1. Oneliner shell calls 3
3.1.1. Database recreation and DDL scripts run one-liners 3
3.1.2. Table(s) load via aa single one-liner 3
3.1.3. Testing one-liner call 3
3.1.4. Test messages user 3
4. RELIABILITY AND STABILITY 3
4.1. Zero tollerance towards crashing 3
4.2. Zero tollerance towards bugs 3
4.3. Daily backups 3
4.4. Logging 3
5. SCALABILITY 3
5.1. Feature scalability 3
5.2. Setup scalability 3
5.3. Projects databases scalability 3
6. PERFORMANCE 3
6.1. Page load maximum time 4
6.2. Login, logout 4
7. MULTI-INSTANCE OPERABILITY AND DEPLOYABILITY 4
7.1. Environment type self-awareness 4
7.2. Cross running between instances of different types 4
8. UI REQUIREMENTS 4
8.1. CRUDs 4
8.1.1. Execution time 4
8.1.2. Visual indication 4
8.2. Clarity on errors 4
9. DOCUMENTATION 4
9.1. Documentation completeness 4
9.2. Documentation and code base synchronization 4
9.3. Full backup to the cloud in less than 5 minutes 4

REQUIREMENTS

1. INTRO

1.1. Purpose

The purpose of this document is to present the requirements set to the Issue-Tracker 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 the application.

1.3. Related documentation

This document is part of the Issue-Tracker application documentation-set, which contains the following documents:
- UserStories - the collection of userstories used to describe "what is desired"
- SystemGuide - architecture and System description
- DevOps Guide - a guide for the developers and devops operators
- Installation Guide - a guide for installation of the application
- Features and Functionalities - description of the current features and functionalities

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

2. DEPLOYABILITY

The issue-tracker must be easily deployable on any Unix like OS. Windows family based OS'es are explicitly out of the scope of the issue-tracker tool. Any issue-tracker 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 should be ready for use in a "blank" OS host in less than an hour.

2.1.1. Deploy by simple unzip

The issue-tracker 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.

2.1.2. 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

2.1.3. Oneliner for Perl modules check

All the required Perl modules, must be verifiable via a single runnable perl script.

2.1.4. 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.

2.2. A full application clone should be ready for less than 5 minutes

A DevOps operator should be able to perform an application clone of the issue-tracker application in less than 5 minutes.

2.2.1. Shell script for postgres db creation

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

2.2.2. One liner for single restore

The full data example of a cloned from the issue-tracker db should be loadable with a single shell call.

3. USER-FRIENDLINESS

The interaction with each endpoint and interface of an application instance should be as user-friendly as possble.
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 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

3.1.2. 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

3.1.3. 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.

3.1.4. Test messages user

Each test should obey 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. RELIABILITY AND STABILITY

4.1. Zero tollerance towards crashing

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

4.2. Zero tollerance towards bugs

All bugs and inconsistencies must be delt with top priority bypassing new features implementation.

4.3. Daily backups

Daily backups should be show-stopper for the normal operation of the application - that is if an instance is to be considered as normally operating , the daily backups should be performed automatically as indispensable part of the functioning of the application.

```
# add the following one-liner in the cron-scheduled script
# load the proj env vars
doParseCnfEnvVars <<path-to-proj-file>>
clear ; export tables=`curl -s -k http://$web_host:3000/$postgres_db_name/select-tables|jq -r '.| .dat| .[] | .table_name'|perl -ne
's/\s+/,/g;print'`;export do_truncate_tables=1 ; export rdbms_type=postgres ; export load_model=upsert ; perl
src/perl/issue_tracker/script/issue_tracker.pl --do json-to-db --tables $tables
```

4.4. Logging

The application shoud support configuratble logging to STDOUT and STDERR for the following levels - debug,info,warn

5. SCALABILITY

5.1. Feature scalability

The addition of new features should be as scalable as possible.

5.2. Setup scalability

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

5.3. Projects databases scalability

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

6. PERFORMANCE

6.1. Page load maximum time

Each page of the application containing less than 2000 items 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 issue-tracker 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 appalication layer should be able to fetch data from a prd database.

8. UI REQUIREMENTS

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 ok messages , but only error messages, yet the UI should be as responsive that the end-user would easily undertand 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 occured, however no msgs should be displayed when the data is saved properly.

9. DOCUMENTATION

9.1. Documentation completeness

Each running instance MUST have the following documentation set :
- Features and Functionalities doc
- DevOps Guide
- Requirements
- SystemGuide
- UserStories document
- Installation and Configuration Guide

in at least the md and pdf file formats.

9.2. Documentation and code base synchronization

Each running instance MUST have its 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.

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

A gull backup of sotware,configuration and data for the issue-tracker and/or another project database should be doable in less than 5 minutes. The backup should be easily searchable from the cloud as well.