

## Table of Contents

Table of Contents	1
ISSUE-TRACKER USERSTORIES AND SCENARIOIS	2
1. BIZ PERSPECTIVE	2
1.1. Switch between projects	2
1.1.1. Current project visibility	2
1.2. Track issues progress	2
1.2.1. Measure success	2
1.2.2. Monitor success	2
1.3. Track issues history	2
1.4. Time management	2
1.4.1. time centric planning	2
1.4.2. time centric reporting	2
1.5. Track issues relations	3
1.6. Access issues txt format from email	3
1.7. Access issues data from Google sheet	3
1.7.1. Apply publish filter while posting to Google Sheet	3
1.8. Project's persons issue combinations	3
2. SYSADMIN PERSPECTIVE	3
2.1. System deployability	3
2.2. System performance	3
2.3. System stability	3
2.4. System reliability	3
2.5. Ease of use	3
3. ETL AND INTEGRATIONS PERSPECTIVE	4
3.1. Xls-to-mysql-db hierarchical data load	4
3.1.1. error reporting in xls-to-mysql-db hierarchical data load	4
3.2. Xls-to-postgres-db hierarchical data load	4
3.2.1. error reporting in xls-to-postgres-db hierarchical data load	4
4. DEVOPS PERSPECTIVE	4
4.1. System verifiability and testability	4
4.1.1. Clarity and breavity of the end to end tests	4
4.1.2. Abort end-to-end tests on single test fail	4
4.1.3. Control flow logging	4
4.1.4. Single entry point for end to end tests	4
4.1.5. Tool run log to human readable description	5
4.1.6. Userstories to test case relations	5
4.1.6.1. UUID tracability for test files and userstories	5
4.1.7. Components start run message print	5
4.1.8. Tool exit with exit code and exit message	5
4.1.9. Execution path tracing by UUID's	5
4.2. Manage details of issues with a single txt file	5
4.2.1. Issues directories naming conventions	5
4.2.2. Issues files naming conventions	5
4.2.3. Issues file open	5
4.2.4. Issues files history	5
4.2.5. Issues files naming conventions	6
4.2.5.1. Issues files naming conventions for the project	6
4.2.5.2. Issues files naming conventions for current date	6
4.2.5.3. Issues files naming conventions for the time frame	6
4.3. Issues transformations	6
4.3.1. Load by txt-to-db action	6
4.3.1.1. Load issues file from file system to db	6
4.3.2. Load issues by db-to-xls action	6
4.3.3. Load issues by xls-to-db action	6
4.3.3.1. Load issues by xls-to-db action for insert or upset	6
4.3.3.2. Load issues by xls-to-db action by truncating or not the loadable table	7
4.3.4. Load issues by db-to-txt	7
4.3.4.1. xls-to-db action load sort by issues prio attribute	7
4.3.4.2. db-to-txt action load sort by issues start_time attribute	7
4.3.4.3. db-to-txt action load sort by issues start_time attribute	7
4.3.5. Load issues file from db to file system	7
4.4. issues file filtering	7
4.5. Single shell call for projects switching	7
4.6. Issues publishing from shell calls	7
4.6.1. Issues publishing in e-mail format	8
4.6.2. Issues handling in google sheet format	8
4.6.3. Issues publishing in google calendar format	8
4.7. Metadata handling	8
5. UI PERSPECTIVE	8
5.1. Performance	8
5.1.1. Page load times	8
5.2. Projects switching	8
5.3. Time management	8
5.3.1. copy an issue-tracker instance issue to a google calendar event	8
5.4. Issues listing	8
5.4.1. automatic issue items sequencing	9
5.4.2. Issues re-ordering by desired or default attribute in list view	9
5.4.3. Issues list default row height	9
5.5. Issues data transfer between different projects	9
5.6. Issues export to Google calendar	9
5.7. Issues import from Google calendar	9
6. UI DEVELOPER PERSPECTIVE	9
6.1. Testability	9

# ISSUE-TRACKER USERSTORIES AND SCENARIOIS

## 1. BIZ PERSPECTIVE

As an issues-manager  
In order to operate successfully my business  
I wanto to be sure that each time unit spent on the tool is business and performance centric.

### 1.1. Switch between projects

As an issues-manager  
In order to manage issues from different projects  
I wanto to be able to switch between different projects easily and quickly

#### 1.1.1. Current project visibility

As an issues-manager  
In order to avoid confusion between different projects  
I wanto to be able to see the current project name from any interface I am working in quickly and esily.

### 1.2. Track issues progress

As an issues-manager  
In order to keep track on what and when was accomplished vs. what was planned  
I wanto to be able to keep track on daily basis what was planned and what whas accomplished on a project daily, weekly, monthly, yearly and decadally issues.

#### 1.2.1. Measure success

As an issues-manager  
In order to measure the success of the planned issues  
I wanto to be able to measure the deliverables of each issue by comparable metrics.

#### 1.2.2. Monitor success

As an issues-manager  
In order to monitor the success of the planned issues  
I wanto to be able to monitor the metrics of the issues.

### 1.3. Track issues history

As an issues-manager  
In order to keep track on what and when was planned on daily basis  
I wanto to be able to keep track what was planned on a project term - day,week,month,year,quinquennial or decade

### 1.4. Time management

As an issues-manager  
In order to be prepared for issues such as ( events , tasks ) which have start and stop time  
I wanto to be able to save their start\_time and stop\_time per issue in every possible interface

#### 1.4.1. time centric planning

As an issues-manager  
In order to be able to plan the issues data for a certain term - day,week,month,year,quinquennial or decade  
I wanto to be able to perform all the features of the issue-tracker on that specific period regardless whether it is today , in the past or in the future

```
bash src/bash/issue-tracker/issue-tracker.sh -a increase-date -d today
bash src/bash/issue-tracker/issue-tracker.sh -a increase-date -d tomorrow
```

#### 1.4.2. time centric reporting

As an issues-manager  
In order to be able to report the issues data for for a certain term - day,week,month,year,quinquennial or decade  
I wanto to be able to perform all the features of the issue-tracker on that specific day regardless whether it is today , in the past or in the future

```
bash src/bash/issue-tracker/issue-tracker.sh -a increase-date -d yesterday
bash src/bash/issue-tracker/issue-tracker.sh -a increase-date -d yesterday
```

---

### 1.5. Track issues relations

As an issues-manager of a software related project  
In order to trace the issues relations to userstories, features and tests  
I want to be able to access the related objects to an issue by means of a link

### 1.6. Access issues txt format from email

As a user of the issue tracker tool  
In order to be able to access and read my issues from a mobile device  
I want to be able to send each period txt file from the daily folder via gmail.

### 1.7. Access issues data from Google sheet

As the biz user of the issue tracker tool  
In order to be able to share and edit the data with multiple users authenticated within the Google eco system  
I want to be able to access , edit and update the issues data from google sheet

#### 1.7.1. Apply publish filter while posting to Google Sheet

As the biz user of the issue tracker tool  
In order to show only relevant data to the future viewers of the published to Google sheets issues data  
I want to be able to apply publishing filter for columns to be left unpublished per item table per project

### 1.8. Project's persons issue combinations

As the project manager of an issue-tracker project  
In order to be able to quickly and reliably combine the reported hours by the project's people  
I want to be able to read their issue-tracker formatted google sheets and combine them into a single project's google issue-tracker sheet

## 2. SYSADMIN PERSPECTIVE

As a sysadmin of the issue-tracker tool  
In order to operate the issue-tracker tool instance on my responsibility efficiently  
I want to be sure that each aspect of my tasks and activities are covered by the functionalities of the tool to the maximum possible extend.

### 2.1. System deployability

As the SysAdmin  
In order to quickly take into use a new product instance of the issue-tracker tool running on a separate host accessible via ssh  
I want to run a single deploy-host action - which will install the required OS libraries , Postgres and Perl modules for the operation of the tool silently.

# <https://serverfault.com/questions/364452/silent-and-scripted-install-of-cpan-and-perl-modules>  
# <https://serverfault.com/a/815650/33346>

### 2.2. System performance

As the SysAdmin  
In order to ensure the performance of the System  
I want the System containing the issue-tracker tool to perform its functions within the defined performance criteria

### 2.3. System stability

As the SysAdmin  
In order to minimize downtimes and ensure continuous operations  
I want the System containing the issue-tracker tool to perform its defined functions on request without interruptions or unknown side effects

### 2.4. System reliability

As the SysAdmin  
In order to be able to rely on the operations of the tool  
I want the System containing the issue-tracker tool to perform its functions as specified consistently

### 2.5. Ease of use

As the SysAdmin  
In order to be efficient and decrease the amount of errors

I want to generally perform any command the system within the sysadmin scope via clean and memorable oneliners

### **3. ETL AND INTEGRATIONS PERSPECTIVE**

#### **3.1. Xls-to-mysql-db hierarchical data load**

As the Data Integrator

In order to be efficient while handling the System's hierarchical data

I want to be able to

use a single shell call to load all or chosen table(s) to the mysql db

##### **3.1.1. error reporting in xls-to-mysql-db hierarchical data load**

As the Data Integrator

In order to be efficient while troubleshooting data loading errors

I want to be able to see :

- which table's load failed

- what was the error in failed to

#### **3.2. Xls-to-postgres-db hierarchical data load**

As the Data Integrator

In order to be efficient while handling the System's hierarchical data

I want to be able to

use a single shell call to load all or chosen table(s) to the postgres db

##### **3.2.1. error reporting in xls-to-postgres-db hierarchical data load**

As the Data Integrator

In order to be efficient while troubleshooting data loading errors

I want to be able to see :

- which table's load failed

- what was the error in failed to

### **4. DEVOPS PERSPECTIVE**

As a devops operator of issue-tracker tool

In order to develop and operate the instance of the issue-tracker tool efficiently

I want to be sure that each aspect of my tasks and activities are covered by the functionalities of the tool to the maximum possible extend.

#### **4.1. System verifiability and testability**

As an ITOPS

In order to be able to rely on the operations of the tool

and manage easily its features and functionalities

I want to easily verify and test parts or the whole System

##### **4.1.1. Clarity and brevity of the end to end tests**

As an ITOPS

In order to be able to verify all the features and functionalities of the tool within the System

I want to see the results of each test in 1 row in the following format:.

##### **4.1.2. Abort end-to-end tests on single test fail**

As an ITOPS

In order to be able to run continuously end-to-end tests and skip for several runs failing tests

I want to be able to configure the single e2e entry point script to skip certain tests, but report me what was skipped.

##### **4.1.3. Control flow logging**

As a CLI user

In order to be able to understand what the issue tracker tool is executing

I want to have configurable logging with stderr, stdout and file output

##### **4.1.4. Single entry point for end to end tests**

As an DevOps

In order to be able to verify all the features and functionalities of the tool within the System

I want to run a single shell call running all the end-to-end test of the application ensuring the prespecified features and

functionalities.

#### **4.1.5. Tool run log to human readable description**

As a CLI user

In order to be able to get a human readable description of the log of the specific run of the tool

I want to be able to translate the recorded uuid's in the execution run log to their respective records

#### **4.1.6. Userstories to test case relations**

As a Developer

In order to ensure the stability and expandability of the application

I want to be able to run for each implemented userstory a single test

##### **4.1.6.1. UUID tracability for test files and userstories**

As a Developer

In order to identify each userstory to be tested with its according test

I want to be able to track each userstory or test code entry point file by UUID.

#### **4.1.7. Components start run message print**

As a CLI user

In order to know when a component has been started

I want to be able to see the "START <<COMPONENT\_NAME>>" on either the STDOUT or the log file of the component

#### **4.1.8. Tool exit with exit code and exit message**

As a CLI user or calling calling automated component

In order to be able to understand whether or not the execution of the call to the tool was successful or not

I want to get the exit code from the tool execution and see the exit message

#### **4.1.9. Execution path tracing by UUID's**

As a DevOps operator

Foreach execution run of the tool

I want to be able to walk trough the execution path of the tool programatically.

### **4.2. Manage details of issues with a single txt file**

As a CLI user

In order to be able quickly to view my issues

I want to be able to update the names , statuses and categories of my daily,weekly,monthly, yearly and decadally issues by simply editing the issues term file

#### **4.2.1. Issues directories naming conventions**

As a issues manager

In order to be able to manage lots of issues from different projects stored in plain txt files

I want to be able to store large quantity of issues txt files by their date on daily, weekly, monthly, yearly and decadally basis

#### **4.2.2. Issues files naming conventions**

As a issues manager

In order to be able to manage lots of issues from different projects stored in plain txt files

and open them quickly

I want to be able to just type the first letter in a text editor supporting select opened file by typing its first letter and jump to that file

#### **4.2.3. Issues file open**

As a CLI user

In order to be able quickly to access my issues ( daily , weekly, monthly , yearly )

I want to be able to view my daily issues by simply opening a single txt file

#### **4.2.4. Issues files history**

As a CLI user

In order to be able quickly to search through old issues  
I want to be able to access old issues files by their date held in their file names from the file system

#### **4.2.5. Issues files naming conventions**

As a DevOps  
In order to be able quickly to access and manage programmatically issues  
I want to be able to guess the file paths of the issues file by their date

##### **4.2.5.1. Issues files naming conventions for the project**

As a DevOps  
In order to be able quickly to switch between different projects  
I want to have the project name of the issues file in its name as the first token as follows:  
<<issue\_tracker\_project>>-issues.<<current-iso-date>>.<<daily|weekly|monthly|yearly>>.txt

##### **4.2.5.2. Issues files naming conventions for current date**

As a DevOps  
In order to be able quickly and programmatically to go back in the history  
I want to have the current registration date in the file name and path  
<<issue\_tracker\_project>>-issues.<<current-iso-date>>.<<daily|weekly|monthly|yearly>>.txt

##### **4.2.5.3. Issues files naming conventions for the time frame**

As a DevOps  
In order to be able quickly and programmatically to go back in the history  
I want to have the current registration date in the file name and path  
<<issue\_tracker\_project>>-issues.<<current-iso-date>>.<< ( daily|weekly|monthly|yearly ) >>.txt

### **4.3. Issues transformations**

As an cli user of the System  
In order to be able to sort the issues according to their attributes  
and edit them in both txt file and xls file  
I want to be able to perform the following loads:  
txt-to-db - to load a txt file with issues to an issues table in db  
db-to-xls - to load a xls file from db table to xls  
xls-to-db - to load a xls file with issues to an issues table in db  
db-to-txt - to load a txt file from db table

#### **4.3.1. Load by txt-to-db action**

As an cli user of the System  
in order to store my issues in structured form to db for further processing  
I want to be able to load any issue file with a single line shell call to a db

##### **4.3.1.1. Load issues file from file system to db**

As an cli user of the System  
- in order to be able to handle issues from different projects  
- and load them to db for further processing  
I want to:  
- pre-set the variables of the project  
- and then load any issue file with a single line shell call to a db  
- and optionally specify the period of the issues file ( daily , weekly , monthly , yearly ) with daily as default

#### **4.3.2. Load issues by db-to-xls action**

As an cli user of the System  
in order to be able to sort and edit my issues in Excel  
I want to be able to unload my issues from one or many tables in the db at once in a single shell call

#### **4.3.3. Load issues by xls-to-db action**

As an cli user of the System  
in order to store my issues in structured form to db for further processing after being sorted in Excel  
I want to be able to load my latest xls file with a single line shell call to a db  
by choosing the names of the tables to load

##### **4.3.3.1. Load issues by xls-to-db action for insert or upset**

As an cli user of the System

in order to insert or upsert my issues in structured form to db for further processing after being sorted in Excel

I want to be able to load my latest xls file with a single line shell call to a db

by choosing the names of the tables to load and specifying upsert by adding the guid column to the xls sheet

#### **4.3.3.2. Load issues by xls-to-db action by truncating or not the loadable table**

As an cli user of the System

in order to store my issues in structured form to db for further processing after being sorted in Excel

I want to be able to load my latest xls file with a single line shell call to a db

by choosing the names of the tables to load

#### **4.3.4. Load issues by db-to-txt**

As an cli user of the System

in order to store my issues in more structurized data format for further procesing

I want to :

- be able to load the issues for a period from the db to a file
- by choosing the names of the tables to load

##### **4.3.4.1. xls-to-db action load sort by issues prio attribute**

As an cli user of the System

during the db-to-txt action load

in order to understand the priority of my issues

I want to :

- be able to specify the order in the issues txt files lines to be based on the prio attribute
- by choosing the names of the tables to load

##### **4.3.4.2. db-to-txt action load sort by issues start\_time attribute**

As an cli user of the System

during the db-to-txt action load

in order to understand the priority of my issues

I want to :

- be able to specify the order in the issues txt files lines to be based on the start\_time attribute ( start\_time could be in the following format YYYY-mm-dd HH:MM in start\_time or HH:MM )
- by choosing the names of the tables to load

##### **4.3.4.3. db-to-txt action load sort by issues start\_time attribute**

As an cli user of the System

during the db-to-txt action load

in order to view the issues by categories

I want to :

- be able to specify the order in the issues txt files lines to be based on the category attribute

#### **4.3.5. Load issues file from db to file system**

As an cli user of the System

in order to store my issues in more structurized data format for further procesing

I want to :

- be able to load the issues for a period from the db to a file
- and optionally specify the period of the issues file ( daily , weekly , monthly , yearly ) with daily as default

#### **4.4. issues file filtering**

As a CLI user

In order to filter quickly my issues

I wanto to be able to show the issues with their categories of only certain status

#### **4.5. Single shell call for projects switching**

As an issues-manager

In order to be able to switch between different projects quickly

I wanto to be able to issue a single shell call for loading a project's configuration and run the issue-handler against this pre-loaded configurtion

#### **4.6. Issues publishing from shell calls**

As a DevOps

In order to be able to quickly share the current issues data in tabular format

I want to be able to issue a single shell call for copying the current items data to a medium by specifying the tables to be published

#### **4.6.1. Issues publishing in e-mail format**

As a DevOps

In order to be able to quickly share the current issues data in email format

I want to be able to issue a single shell call for copying the current items data to email by specifying the tables to be published

#### **4.6.2. Issues handling in google sheet format**

As a DevOps

In order to be able to quickly share the current issues data in tabular format

I want to be able to issue a single shell call for copying the current items data to google sheet by specifying the tables to be published

#### **4.6.3. Issues publishing in google calendar format**

As a DevOps

In order to be able to quickly share the current issues data in google calendar format

I want to be able to issue a single shell call for copying the current items data to google calendar by specifying the tables to be published for the items having set start\_time and stop\_time attributes.

### **4.7. Metadata handling**

As a DevOps

In order to be able to programmatically manage all aspects of my data

I want to have a single entry point to manage the meta data per tables , columns and UI elements

so that even a table, column or whatever object is not populated in the meta still there will be default values for it usable by the application

## **5. UI PERSPECTIVE**

As an UI user of the issue-tracker tool

In order to manage my issues via the UI successfully

I want to be sure that each aspect of my tasks and activities which could be performed via the UI of the tool are covered by the functionalities of the tool to the maximum possible extend.

### **5.1. Performance**

As an UI user of the issue-tracker tool

In order to enjoy the usage of the tool and interact efficiently

I want to have responsive and quick UI.

#### **5.1.1. Page load times**

As an UI user of the issue-tracker tool

In order to enjoy the usage of the tool and interact efficiently

I want to have a maximum page load time on efficient network less than 0.5 seconds and preferably even 0.3 seconds

### **5.2. Projects switching**

As an issue-tracker ui user

In order to be able to quickly switch between projects

I want to be able to access a web page providing autocomplete to preloaded configuration entries for the different projects

### **5.3. Time management**

As an issue-tracker ui user

In order to be able to prepare for issues such as ( events , tasks ) which have start and stop time

I want to be able to view the issues with the same title, start\_time and stop\_time in google calendar

#### **5.3.1. copy an issue-tracker instance issue to a google calendar event**

As an issue-tracker ui user

In order to be able to see my issues time-schedule via phone and browser in a calendar view

I want to be able to copy via the ui an issue as a new google calendar event

### **5.4. Issues listing**

As an UI user of the system I want to be able to list the issues stored in it.



#### **5.4.1. automatic issue items sequencing**

As an UI user of the system

In order to save time while arranging all the different issue items

I want the System to automatically sequence each item in list view by a default incremental sequence unless I have specified my own sequence.

#### **5.4.2. issues re-ordering by desired or default attribute in list view**

As a UI user

In order to prioritize and re-arrange to a logical sequence my issues

I want to be able to drag and drop issues up and down ,  
which would correspondingly increase or decrease their attribute to which they are currently sorted or ordered by.

#### **5.4.3. issues list default row height**

As a UI user

In order to quickly comprehend the data in the lists

I want each row of the ui to have a certain minimum height and whenever the data cannot fit into this height to be greater than it

### **5.5. Issues data transfer between different projects**

As the UI user of an issue-tracker instance

In order to save be able to track my personal time usage between different projects and the different interdependancies

I want to be able to move issues data from one project to another via the UI

### **5.6. Issues export to Google calendar**

As the UI user of an issue-tracker instance

In order to be able to visualize and manage my start- and stop\_time having issues better

I want to be able to export my issues to Google calendar

### **5.7. Issues import from Google calendar**

As the UI user of an issue-tracker instance

In order to be able to visualize and manage my my start- and stop\_time having issues better

I want to be able to import my Google calendar issues into my issue-tracker profile on an issue-tracker instance

## **6. UI DEVELOPER PERSPECTIVE**

As the UI Developer

In order to be able to deliver working solutions for the UI

I want to have user friendly development experience.

### **6.1. Testability**

As the UI Developer

In order to deliver working ui units

I want to be able to quickly setup the existing project with minimalistic default set of data.