|  |  |
| --- | --- |
| Module | rDQ User Interface |
| Sub-Module | UI Skeleton |
| Author | D Twaddell |
| Date | 19th Aug 17 |
| Reviewer |  |
| Review Date |  |
| Review Status |  |
| Review Comments |  |

# Design Summary

## Design Log

## Sub-Module Description

**UI Skeleton** – Build a skeleton user interface that a user can use to interface with rDQ functions. Use the R Shiny package to implement. Use layout suggested below as the starting point, but we will consider improvements to the design and layout as we go.

Commit the UI to Github so it can be reviewed by the team: <https://github.com/archanalytics/rDataQuality>.

## Sub-Module Context

Enabler for the user interface.

## Data Model

### Database tables for this module

(See rDQ Data Model design document)

**sysParameters** – stores information about parameters used in the system

**rdqModule** -reference data, list of rDQ modules and sub-modules

# Design Details

## Navigation System

Shiny allows for Pages each with different context menus. Likely we should use a separate Page for each Sub-Module. Then each module needs a two-level navigation system, so overall we will have three levels of navigation.

1. rDQ Analyser
   1. Explore data - select connection, browse on the connection, and list available data files. Run table and attribute level analysis to produce interactive and storable and printable profile)
2. rDQ Rules
   1. Data Dictionary - Define data concepts and elements in ‘business’ terms
   2. Rule generator - using a table profile, suggest data quality rules. Allow association of rules to the table definition).
3. rDQ Monitor
   1. Define Control Point – define a control (a set of DQ rules to be applied to a dataset)
   2. Run Control – run rules against the data and generate DQ results. Run manually (later will be able to define an automation schedule)
4. rDQ Transform
   1. TBA
5. rDQ Issues
   1. List Issues
   2. Create / Delete Issue
6. rDQ Ref Data
   1. Update reference tables, and define custom reference data
7. rDQ Dashboard
8. rDQ Reports
   1. Define report
   2. Run report
9. rDQ Administration
   1. User Preferences (see below)
   2. System Log

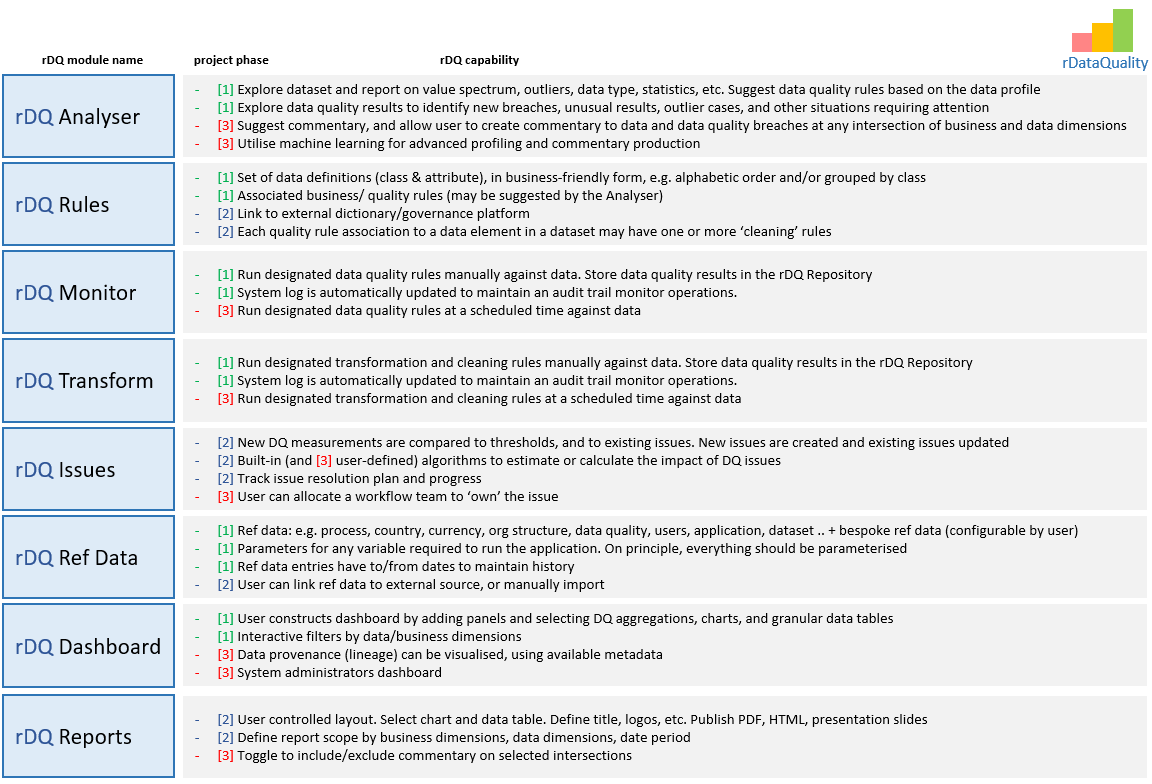
## User Preferences

There should be a link to allow the user to set certain preferences and system parameters. This will be an extensible list so we will add to it as we go. A key development principle is that nothing is ‘hard-coded’, everything is accessible by configurable parameters. The parameters will need to be organised, initial suggestion:

* User Profile
  + Name
  + Organisation
  + Team
  + Telephone
  + Email
  + Image (for UI, and for reports) – request size: 100 x 100 pixels (or shrink user image to these proportions)
* User Preferences
  + Include contact details on reports? Y/N
* System Configuration
  + R path
  + Database connection
    - Database type (SQLite, mySQL, HIVE)
    - Connection string (includes user name and other connect details) – “Test connection”
  + Input paths
    - Analyser
    - Data dictionary
    - Business Rules
    - Reference Data
  + Output paths
    - Data export
    - Reports

Individual modules will have context sensitive parameters, and these should be stored in the same parameter file.

# Appendix 1 – rDQ Module List



In addition to these ‘public’ modules we may need some private modules to help organise work, e.g.:

* **rDQDataManagement** – common functions relating to data management, dealing with files and database operations. Define functions that are used in other modules. Define global variables
* **rDQUI** – implements the user interface, using Shiny