Developer’s Guide for QualityCore

**Table of Contents**

[Purpose and Scope](#_q1j4g5itkwme)

[References](#_wdsly3u558b5)

[Source Code Repository - Git](#_h8emcxvrouhl)

[Use of the Gitflow Branching Model](#_g8f0e9c5skbi)

[Feature Branches](#_fec3nrm7xfzx)

[Hotfixes for Operations](#_jiyzchd3c6k)

[PHP Coding](#_mrmt9i92oqjr)

[Coding Standards](#_ljmygice8dw2)

[PHP Coding Standards](#_ivhjx5d8veix)

[PHPDoc](#_niekuq36wfk)

[Symfony Coding Standard](#_k2abdwaxpcl1)

[Code Comments](#_2pkobmbke3qy)

[Comments About Conditionals](#_mutv84luqnd2)

[Indenting SQL in PHP](#_gkbtthl1sydf)

[Application Logging](#_lo7a7lbxvqbk)

[Error Handling](#_wi2edjt43oq4)

[Calls to die()](#_g1k2e2826ns0)

[QualityCore Programming](#_rcmyv7e05bnu)

[School Testing Windows](#_jcpj8gz1hezo)

[Coding with STWs](#_fytkpera9jib)

[Obtaining STW IDs](#_3qdnn2y8tgpw)

[Class Methods](#_3acy573lq3jr)

[getStwById()](#_mgswcw3ioqfy)

[stwIdToNames()](#_mrc1ti5l6139)

[formatStwString()](#_b1kbzx5743ls)

[Database Management](#_k20o7spnx1b6)

[Database Imports and Exports](#_q1bxt4na7xgn)

[Qc-docker Imports and Exports](#_sct0t2e4ev3l)

[Important details](#_z10rhiwj8psv)

[Best practices for using DB backups](#_r81phcog853h)

[Why are we doing this](#_31lhtq7rehej)

[Tools](#_5cy8evrkzzd4)

[Vim](#_zfngg3k8c751)

[Application Testing](#_cwrkbg1wlenj)

[QC Admin](#_h289a1buiprx)

[Development Team Topics](#_pwel3fjv2qg5)

[Sprint Retrospectives](#_cmbv1glyet4a)

[Sprint 8 - Holidaily](#_y3vgygpefm9k)

[Sprint 7 - Golden City](#_bntblth7wq3z)

[Sprint 6 - Funkworks](#_yzrf1tgwo44p)

[Technical Roundtable 2 - 160818](#_ujbirr8i17h1)

[Review Topics](#_gry8tf7j1y6z)

[Git Pull Requests](#_qdw20sheeyri)

[Qc-docker Database Imports and Exports](#_ofh3rimcs3rq)

[Team Communication](#_czx82ksdxxaf)

[Slack versus Email](#_1lfp2cnm1j4m)

[QC Admin Augments Branch for Developer Tools](#_ir5wg1ziagdb)

[Technical Roundtable 1 - 160727](#_kd0dyridzuji)

[Git Flow Branching](#_gvd9jm4pge5t)

[Coding Standards](#_fnef2hqxlpef)

[Function Return Values](#_90if9q3r2pkb)

[Changing Data Using Pass-by-reference](#_4tneraan8g0m)

[SQL Naming Conventions - Databases and Tables](#_fnlkm0c1g1n0)

[Recent Code Implementations](#_4k6a6l41wqmm)

[QC Admin - New User Menu Generation Method](#_9klva4i30420)

[US10947 - PPMS Course Availability Import](#_p4o11rvamslt)

[Documentation Standards](#_pqwdlqjbnqxz)

[New Document - Developer Story Release Notes](#_k41f6bi4bumt)

[Document Structure Essentials](#_fdo7zhkhp1fy)

[Appendix](#_73a9qjdgxdcj)

[Document Revision History](#_zas7kp547vat)

# Purpose and Scope

This is a team document, so please freely add your content (and update other’s content respectfully). And, update the appendix *Document Revision History* table as appropriate (i.e. significant adds/updates).

# References

The following references are related to this document:

* *Developer Environment Setup* - QualityCore document
* *Developer Story Release Notes* document
* *PPMS Developers Guide - QualityCore* document

# Source Code Repository - Git

## Use of the Gitflow Branching Model

The QualityCore development should closely follow the Gitflow branching model.

The following webpages provide a great resource:

* <http://nvie.com/posts/a-successful-git-branching-model/> - a concise description of the Gitflow model
* <http://danielkummer.github.io/git-flow-cheatsheet/> - a tutorial on essential Gitflow commands

More details of the basic steps follows (as outlined by Marty).

### Feature Branches

1) we will create a branch per story

2) have that branch QA'd/tested

3) when QA is done testing with the branch and has OK'd the work we will close/complete the branch which will merge it back into​\*develop\*​

4) develop can be treated like integral as in creating a container with all the current code that has been merged

a ) this means that branch US123 is closed so any bugs found at this stage ​\*must\*​ be defects

5) once we are ready to perform a release we will create a release branch off of develop for final testing

a ) this is really where the old integral would have fit in but because of dynamic containers we can piece meal this before hand

6) once the release branch is complete a release candidate (this is where staging comes in) will be created for final testing

7) and once final testing is done we can release to prod

### Hotfixes for Operations

1) we will create a hotfix branch off of master

2) we will do our code changes / etc in this branch and do the normal QA in dynamic containers

3) have that branch QA'd/tested

4) when QA is done testing with the branch and has OK'd the work we will close/complete the branch which will merge it back into​\*master\*​

5) we will create a release candidate for final testing (staging again)

6) once final testing is complete we will release to prod

The reason we want the develop owner of a story to close that branch whenever possible is because even though they released something to you for testing other developers won't always know if that developer owner made any more commits that they mean to have QA test after the current tests are complete and this will prevent (hopefully) another developer from accidently releasing untested code.

# PHP Coding

## Coding Standards

### PHP Coding Standards

The PSR coding standard documentation is here:

* <https://github.com/php-fig/fig-standards/tree/master/accepted>

#### PHPDoc

As time permits, the *PSR-5 PHPDoc* standard should be used to comment the PHP code.

Programming Constructs which MAY be preceded by a *DocBlock*:

* file
* require(\_once)
* include(\_once)
* class
* interface
* trait
* function (including methods)
* property
* constant
* variables, both local and global scope.

A complete example could look like the following example ([taken from proposed FIG standard](https://github.com/phpDocumentor/fig-standards/blob/master/proposed/phpdoc.md#55-examples)):

/\*\*  
 \* This is a Summary, which may be one or two lines, ending with a period.  
 \*  
 \* This is an (optional) Description. It may span multiple lines  
 \* or contain 'code' examples using the \_Markdown\_ markup  
 \* language.  
 \*  
 \* @see Markdown  
 \*  
 \* @param int $parameter1 A parameter description.  
 \* @param \Exception $e Another parameter description.  
 \*  
 \* @\Doctrine\Orm\Mapper\Entity()  
 \*  
 \* @return string  
 \*/  
function test($parameter1, $e)  
{  
 ...  
 return $date

}

Complete PHPDoc information can be found at these links:

1. <https://github.com/phpDocumentor/fig-standards/blob/master/proposed/phpdoc.md>
2. <https://en.wikipedia.org/wiki/PHPDoc>

### Symfony Coding Standard

* <http://symfony.com/doc/current/contributing/code/standards.html>

Symfony follows the standards defined in the PSR-0, PSR-1, PSR-2 and PSR-4 documents.

### Code Comments

Most code comments are optional and included at each developer’s discretion.

#### Comments About Conditionals

// comment regarding what the conditional handles

If ( $conditionalTest ) {

// comment regarding what this block handles

} else {

// comment regarding what the else block handles

}

### Indenting SQL in PHP

The objective is to clarify the SQL code for ease of readability.

The from clause is indented 2 spaces. The where clause is indented 2 additional spaces. Any additional lines to the where clause are indented 2 spaces further.

For example:

$sql = "SELECT t2.`sd\_directions` AS sessionDirex, t2.sd\_source\_file AS`sessionDirexAudio

FROM student\_sessions t1, session\_directions t2

WHERE t1.s\_id = ?

AND t1.tfs\_id = t2.sd\_tfs\_id

HAVING …

ORDER BY ...";

Note that the superfluous backticks are not preferred, unless they are required due to whitespace column names (gak!).

## Application Logging

Use PSR-3 standard…

## Error Handling

### Calls to die()

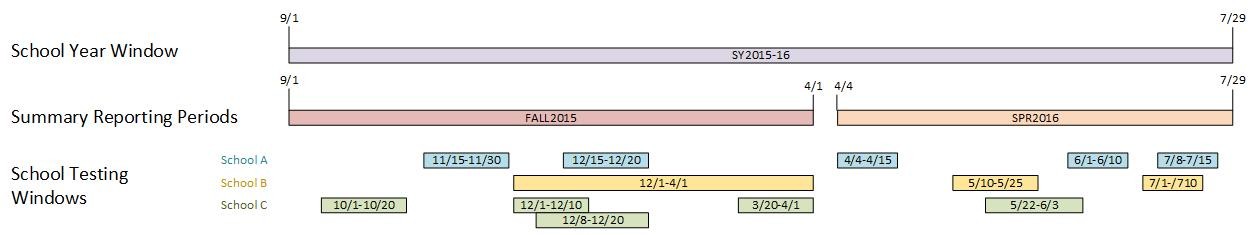
## QualityCore Programming

### School Testing Windows

The QualityCore *School Testing Window* (STW) is a new concept that was introduced in QualityCore 2.0 release.

School testing windows have a date range and an associated name. Specific STWs are unique to a school.

The following diagram depicts an example of how STWs relate to a *School Year Window* and *Summary Reporting Periods* (although the STW names are not shown here):



Note, that in the above example, school testing windows for a given school can overlap.

#### Coding with STWs

Include the STW class into the page’s *configuration module*; for example:

**require\_once** '../../SchoolTestingWindows/class.SchoolTestingWindows.php';

#### Obtaining STW IDs

Obtaining the school testing window ID can be as simple as modifying an existing query. For example:

// query for student tests with all sessions completed,

// obtaining STW ID via student\_sessions->roster->teacher\_class->school\_testing\_windows

// left join to school\_testing\_windows in case a past test associated by teacher\_classes didn't get modified

// with a STW

$sql = "/\* class.VoidStudentTest.php \*/

SELECT DISTINCT tf.tf\_id, tf.tf\_name, tf.tf\_external\_id, tf.tf\_num\_sessions,

sum(ss.test\_session\_complete) as sessions\_completed, teacher\_class.school\_testing\_window\_id,

school\_testing\_windows.display\_name, school\_testing\_windows.start\_date, school\_testing\_windows.end\_date

FROM student\_sessions ss INNER JOIN test\_forms tf ON ss.tf\_id = tf.tf\_id

join roster on roster.r\_id = ss.r\_id

join teacher\_class on teacher\_class.tc\_id = roster.r\_tc\_id

left join school\_testing\_windows on school\_testing\_windows.id = teacher\_class.school\_testing\_window\_id

WHERE ss.st\_au\_id = {$st\_au\_id}

GROUP BY tf.tf\_id, tf.tf\_name, tf.tf\_external\_id

HAVING tf.tf\_num\_sessions = sessions\_completed";

$res2 = $this->dbc->sendQuery($sql);

Again, a STW ID could be easily obtained via a query; for example:

$select = "/\* class.AssignClass.php \*/

SELECT tc\_course\_description, tc\_period, tc\_course\_type, tc\_night\_class,

tc\_id, tc\_au\_id, tc\_course\_code, tc\_section, tc\_testing\_type,

school\_testing\_window\_id

FROM teacher\_class

WHERE tc\_au\_id = {$tc\_au\_id}

ORDER BY tc\_period, tc\_course\_type, tc\_course\_description ASC";

#### Class Methods

The class methods described in this section is not a complete list and may not reflect a current state of the class code.

##### getStwById()

To get all of the STW attributes, from a database row, use the getStwById() method.

For example:

$stwOperator = **new** SchoolTestingWindows($dbc);

$stwRow = $row['school\_testing\_window\_id'] ? $stwOperator->getStwById($row['school\_testing\_window\_id']) : **null**;

$stwNames = $stwRow ?

$stwOperator->formatStwString($stwRow['start\_date'], $stwRow['end\_date'], $stwRow['display\_name']) :

["full" => "Unassigned"];

$table .= “...

<td><b>Testing Window:</b>&nbsp;{$stwNames['full']}</td>”;

##### stwIdToNames()

To get a formatted STW name string, from a STW ID, use the stwIdToNames() method.

For example:

$stwOperator = **new** SchoolTestingWindows($dbc);

$stwNames = $stwOperator->stwIdToNames($row['school\_testing\_window\_id']);

$stwRow = $row['school\_testing\_window\_id'] ? $table .= “...

<td><b>Testing Window:</b>&nbsp;{$stwNames['full']}</td>”;

##### formatStwString()

If a query can return the STW columns, then formatting a STW string is easy.

// create student test dropdown

**while** ($row2 = $res2->fetchRow()) {

$stwName = $row2['display\_name'] ?: "Unassigned"; // null STW ID means unassigned

$stwStartDate = $row2['start\_date'] ?: "-";

$stwEndDate = $row2['end\_date'] ?: "-";

$formattedStwStrings = SchoolTestingWindows::*formatStwString*($stwStartDate, $stwEndDate, $stwName,

SchoolTestingWindows::*FORMAT\_STW\_STRING\_STANDARD*);

$testOptionHtml .= "\t<option value='{$row2['tf\_id']}' title='{$formattedStwStrings['full']}'{$selected}>{$row2['tf\_name']} ({$row2['tf\_external\_id']}) ({$formattedStwStrings['short']})</option>\n";

# Database Management

## Database Imports and Exports

### Qc-docker Imports and Exports

There has been some concerns around docker space and as part of this some confusion on were DB backups should go and how they should be handled, here are the basics of how we need to handle DB backups.

#### Important details

The [qc-docker.pacificmetrics.com](http://qc-docker.pacificmetrics.com/) host has a DB backup storage directory which is:

* /data/mysql\_container\_dev\_backups/

The mysql container can access this directory from by using it's:

* /db\_backups

#### Best practices for using DB backups

1) check the DB backup storage location to see if the file you need already exists

2) if it doesn't exist feel free to use your home directory while performing the copy/scp but once the copy is complete move the file into the DB storage location and out of your home directory

3) be aware that the DB storage location is already accessible from inside the mysql container so you do not need to copy these DB backup files into the container

4) when creating a backup of a database from inside the container the backup should be sent directory to the containers link to the DB backup location

a) it's not the end of the world if you start a backup and it goes to the containers root directory (the way we've always done it) but once the backup is complete this file should be moved

#### Why are we doing this

1) the container has a max size of 10Gb and having the various backups stored in the container was creating various issues with an extract process used by QA/research

2) having a single shared location to store DB files will make it easier for us to maintain these backups and prevent having duplicate files in various locations wasting space

3) keeping the files in a directory on the host will make it easier to maintain and troubleshoot disk space issues

# Tools

## Vim

An example of some useful vimrc settings are:

" .vimrc for sheffel

" MODIFICATIONS:

" 160202 Sheffel Add settings

:colorscheme torte

set number

set tabstop=4

set shiftwidth=4

set expandtab

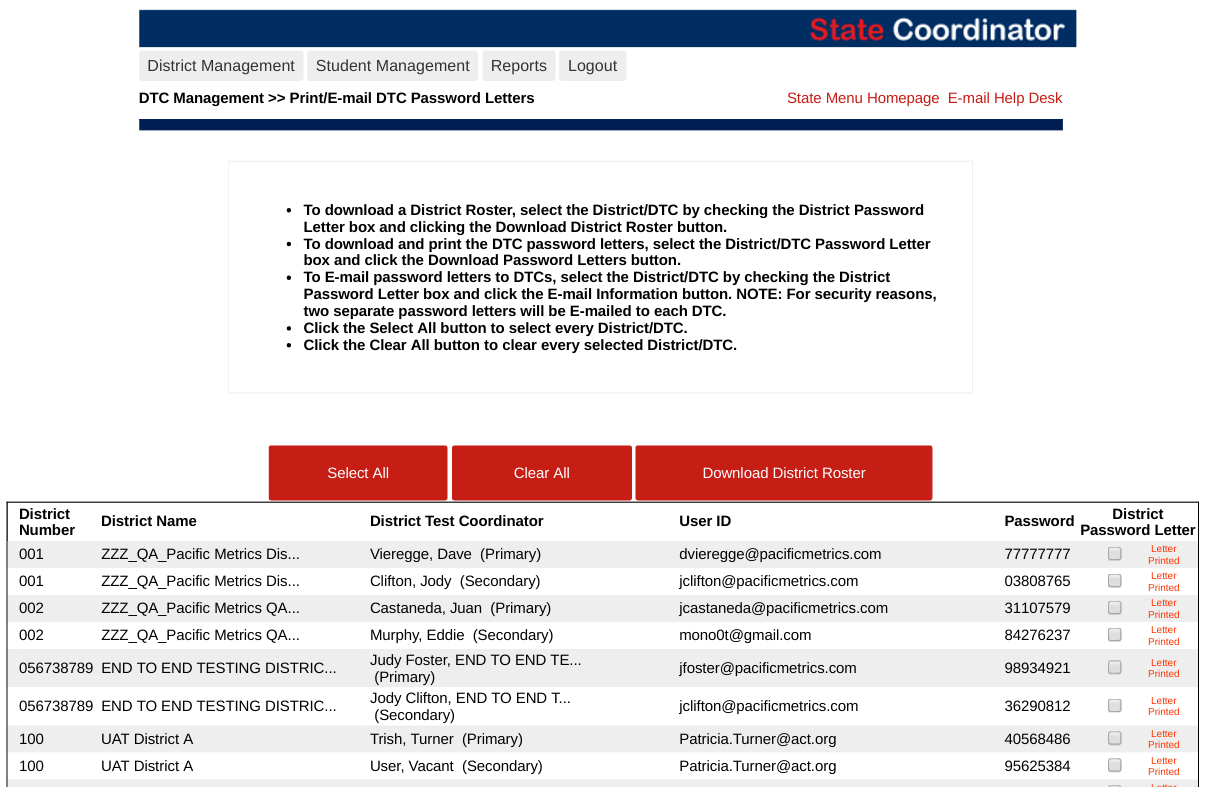
# Application Testing

## QC Admin

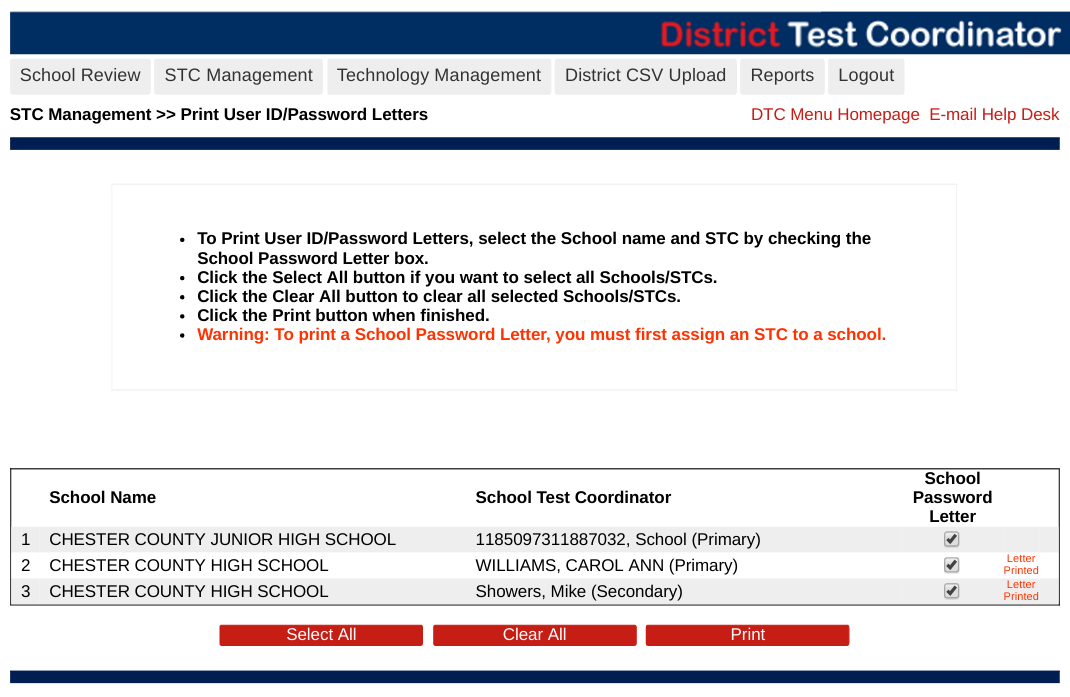
This section needs to be completed (jsheffel).

Log in using dvieregge with 8888 password

Use the DTC Management -> Print/Email DTC Password Letters, as shown below



Login as an STC, using the user ID and password (listed above)



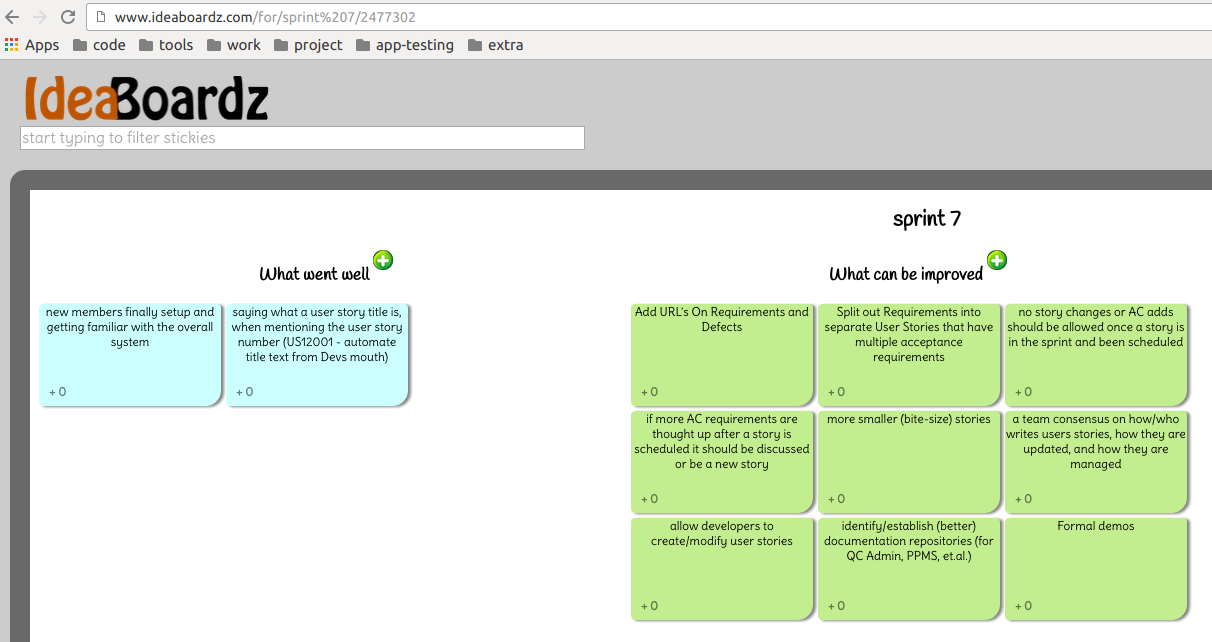
Print and view the resulting user ID and passwords.

# Development Team Topics

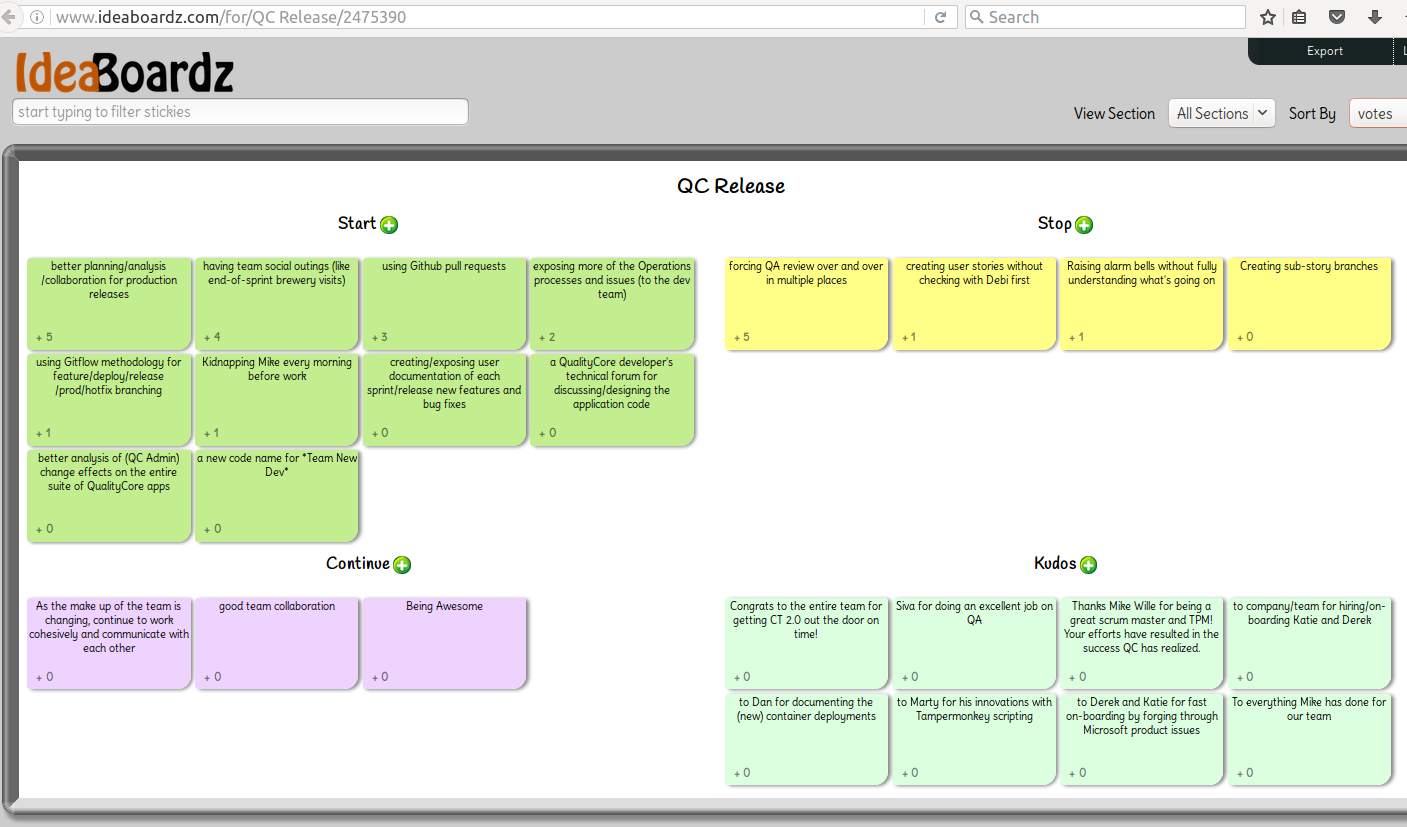
## Sprint Retrospectives

### Sprint 8 - Holidaily

### Sprint 7 - Golden City



### Sprint 6 - Funkworks



## Technical Roundtable 2 - 160818

### Review Topics

* Retrospectives and Other Meetings

It seems that the sprint retrospective meetings have been obsoleted; hopefully not the brewery retro meetings (of which, we have yet to attend, <sad-face>)

* Rally Agile Central - the current state of team use

As a team we should enforce better use of Rally and ensure user stories follow best practices; including splitting long stories and making dynamic updates.

### Git Pull Requests

The *New Dev* team has been talking about using the *Git pull requests* feature. How can we make this happen?

Marty has been *voluntold* to test the pull request capabilities, by using the QC Admin *augments* branch, and requesting a merge into the *develop* branch. Results and recommendations to follow.

### Qc-docker Database Imports and Exports

Marty documented Docker database imports and export, within a Slack channel. Sheffel then copied his text to this document; see the new *Database Management* section.

### Team Communication

#### Slack versus Email

Most team and cross-team communications are occurring in various Slack channels, which is fantastic. Yet, occasionally, an email thread begins. What are the pros and cons?

* Slack provides useful features, like search of all past messages, and that all users (of a given channel) can see all messages (ie. nobody is excluded). But, in Slack, multiple threads get intermixed, which causes difficulties.
* Yet, emails should be used for very official communications, because it (almost) guarantees all recipients will see the message, and there will be a solid thread of conversation.
* In some instances, threads will move from email, to Slack, and back to email.

### QC Admin Augments Branch for Developer Tools

All developers should be aware of the QC Admin augments branch (which should be merged into the develop branch soon).

## Technical Roundtable 1 - 160727

### Git Flow Branching

Keeping branches simple and easily/quickly deployable

* Merging develop back into a feature branch
* Do not merge non-deployable code; talk to team

Keep branches updated with the branch source (features => develop | hotfixes => master)

* Again, merge develop often

Walkthrough creating a release branch and the overall build process

* See Git Flow online documentation

### Coding Standards

#### Function Return Values

list($rc, $data) = getStwId( $array );

getStwId(); // void context

#### Changing Data Using Pass-by-reference

Sometimes necessary but comments to make it clear would be nice to haves..

function addElement( array &$data, array &$array2 );

A valid pattern. Above function updates the two data arrays.

Use good comments.

#### SQL Naming Conventions - Databases and Tables

Use underscores and not hyphens in names.

`my-table` must be escaped.

### Recent Code Implementations

#### QC Admin - New User Menu Generation Method

See upcoming new user story.

Dynamic, recursive to generate breadcrumbs, has management page to ease development, need to workout JS bugs (hyperlink embed issues).

#### US10947 - PPMS Course Availability Import

Refer to the (new) *Developer Story Release Notes* document (sorry Marty… had to do it).

### Documentation Standards

#### New Document - Developer Story Release Notes

Simply communicates a developer’s notes, on a user story, to other developers and QA testers.

#### Document Structure Essentials

Insufficient documentation structure

* Examples of insufficient documentation structure
* Example of sufficient documentation structure

Uh, this document, hello?

# Appendix

## Document Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Modification** |
|  |  |  |
| 160818 | jsheffel | Add *Technical Roundtable Meeting 2* section/notes; |
| 160810 | jsheffel | Add *Database Management* section; add Marty’s qc-docker database import/export content (from Slack channel); |
| 160722 | jsheffel | Add *Sprint Retrospectives* section; |
| 160721 | jsheffel | Add *Development Team Topics* section with first *Roundtable* outline; |
| 160711 | jsheffel | Add Marty’s brief write-up of using the Git Flow model (features and hotfixes); |
| 160628 | jsheffel | Add Source Code Repository section and Gitflow subsection; |
| 160624 | jsheffel | Add *References* section; add school testing windows diagram and explanation; |
| 160610 | jsheffel | Add *PHPDoc* section; |
| 160607 | jsheffel | Add *School Testing Windows* programming section; |
| 160506 | jsheffel | Add *Code Comments* section; |
| 160505 | jsheffel | Add sections: Purpose, Coding Standards |
| 160504 | jsheffel | Initial version |