QA PROCESS & PROCEDURES

AUTHOR(S):

Carl Arndt – QA Manager
Carl.arndt@fcb.com

DATE: 06/29/2016
VERSION: 1.1

FCBCHICAGO

0 INDEX

1 DOCUMENT CONTROL

Document location

Location
\\Chidrffls04\qa\Quality Assurance Process Docs

Author

Name	Position	Contact
Carl Arndt	QA Manager	Carl.arndt@fcb.com

Revision history

Version	Date	Author/editor	Description/Summary of changes
1.1	06/29/2016	Carl Arndt	Various updates
1.0	07/27/2015	Carl Arndt	Initial document creation and styling.

Reviewed by

Version	Name	Position	Review date

Approvals

Version	Name	Position	Approval date

Related documents

Document	Location

2 OVERVIEW

2.1 INTRODUCTION

The FCB Chicago Digital Department contains the Quality assurance Department. The QA Department's primary responsibility is ensuring that the digital department (as well as contractors and sole proprietors doing development work on FCB's behalf) is delivering quality products that meet client requirements and expectations.

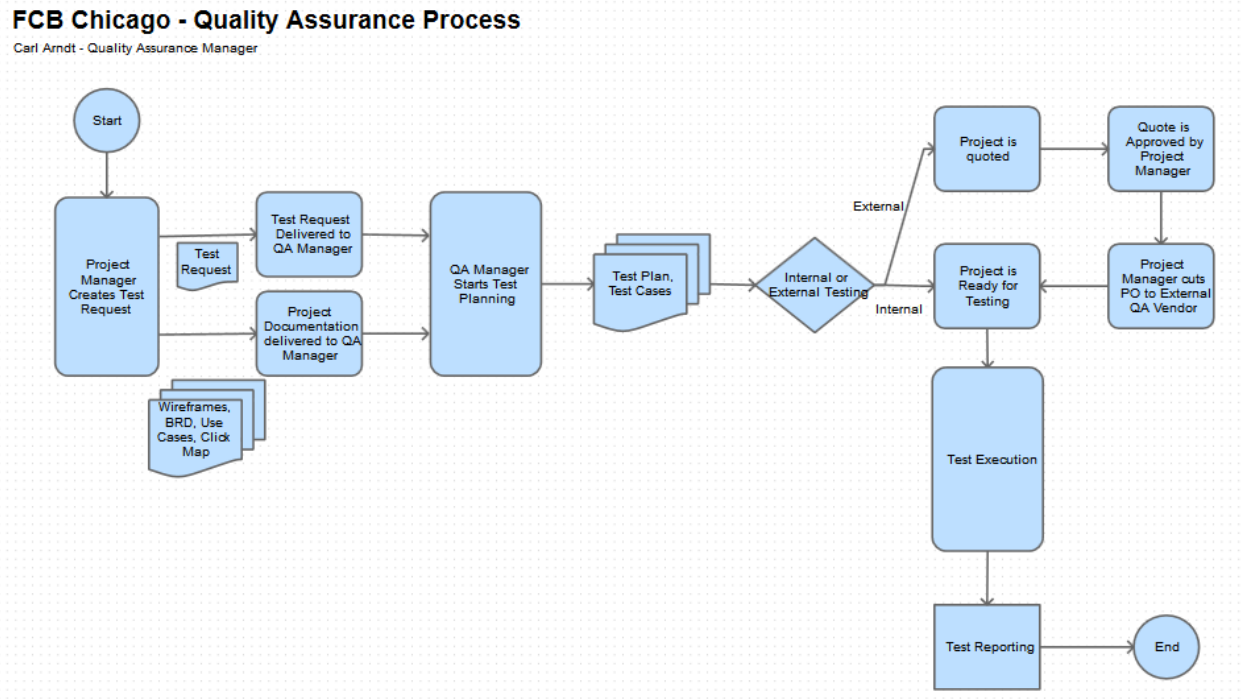
QA follows established processes and methodologies to ensure these quality products are delivered.

2.2 PURPOSE

The purpose of this document is to describe what and how these QA process are and how to follow them. This document will be especially useful to new team members joining the project and need to know what quality expectations are, how the QA team tests, and what processes are in place (and what is being developed for future projects).

This document is for anyone that works with any area of Quality with the FCB Digital Department. All readers, from Executive Managers, Account representatives, Project Managers, Developers, and other Quality Assurance testers in consulting capacities, will gain a perspective of how QA functions in the Software Development Life Cycle.

3 QUALITY ASSURANCE OVERALL PROCESS



The above diagram illustrates the high-level picture of how the Quality Assurance process functions at FCB Chicago. This document will be diving into the details and particulars of each section in the coming pages.

4 TEST PLANNING

4.1 TEST REQUEST

Before any test cases are executed, and before any websites, emails, or banners are tested, test plans are first created. The first action item that kicks off the QA process is the Test Request.

To engage QA and start the testing process, a Project Manager creates a Test Request ticket in the Jira Issue Tracking System. The Test Request Ticket will contain all necessary information in order to start planning for and testing the project under test.

The following information is entered into a Test Request:

a. Summary – One line description of the project that will be tested
b. Description – Description of project, links, dates of testing, In and Out of Scope, etc…
c. Platforms – Platforms this project will support (please include version numbers)
d. Browsers – Browsers this project will support (please include version numbers)
e. Job Code – Approved job code for the project
f. Attachments
g. Wire-frames – The framework of how the site/email is structured goes along ways in test planning
h. BRD (Business Requirements Documents) – There are many reasons why a project needs a well-documented BRD. Some of those reasons include that:
i. QA needs to know exactly how the system is intended to function so we know how to write Expected Result values for Test Cases
j. QA needs to be able to map requirements in the BRD with Test Cases in the Test Run. More information on Requirements Mapping to come.
k. Use Cases – Spells out how the app should function
l. Click Maps – On click, links are sending the user to the intended location

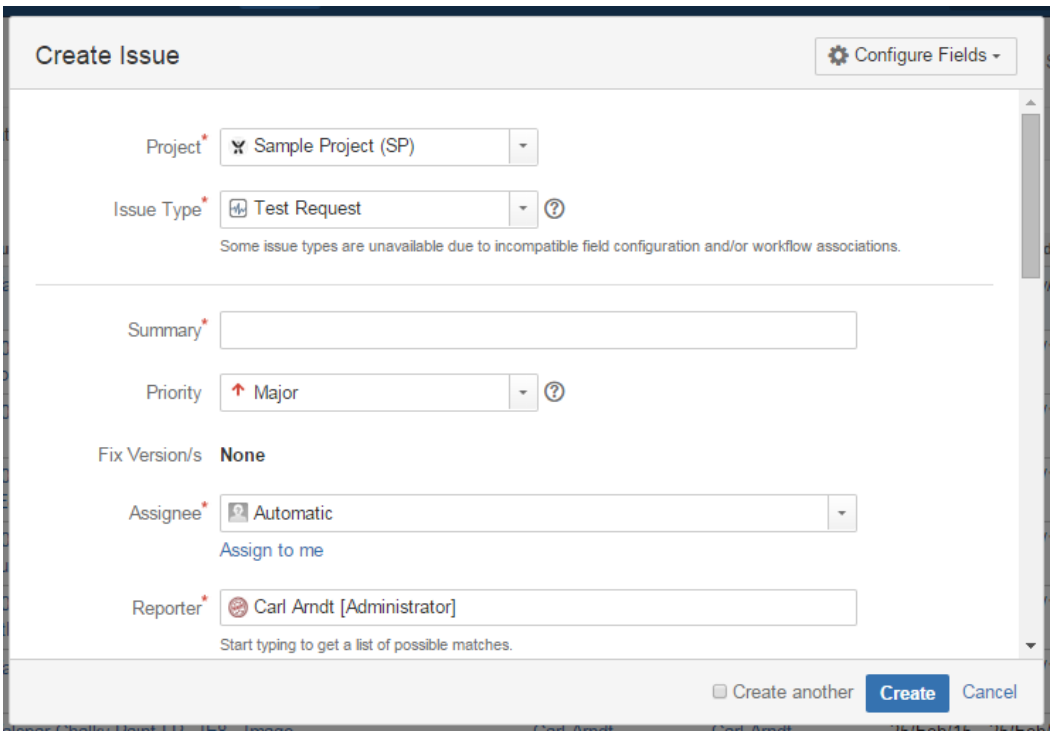
It's worth mentioning here that not all of this information is required for QA to start planning for the testing effort. Any of the above information that is delivered to QA with **as much prior notice as possible** before testing time is very helpful (and necessary) for QA. Additional information can be entered at a future time to the test request as it becomes available.

In Jira, the Test Request issue type is available on all projects. Any user that can access the system can create them.

To create a Test Request:

1. Log into Jira at https://draftfcbdev.atlassian.net/
2. Click the Blue Create button at the top of the screen
3. Choose a Project

4. Choose 'Test Request' for issue type
5. Enter a brief description of the testing needed to be done in the Summary field
6. Complete the remaining required fields
7. Assign the ticket to yours truly (Carl Arndt (Administrator))
8. Submit ticket



Create Issue window in Jira

4.2 PROJECT LEVEL PLANNING

Project Quoting and Estimation are in the Project Level Planning phase. This work is performed in order to provide Project Management a picture of what the costs and level of effort would be if the QA Team were to work on the project.

4.2.1 ESTIMATION

The QA Manager is responsible for estimating the time and costs of a testing effort. Once the PM requests a Quality Assurance testing estimate, the QA manager provides one using the Estimation template (QA Estimate Template 3.1.xls). Hours are divided on the template by test management and test execution hours.

4.2.2 QUOTING

The QA department works with QA vendors outside of the FCB organization. Requests to the QA vendors are sent, and they return quotes on their portion of the work for the project. These quotes can also be included when needed in the QA Estimate document (above) as part of the whole QA Estimate.

4.3 TEST PLAN

The Test Plan is the document that encompasses everything related to testing for a given project, and in some cases, a group of like projects. Medium sized projects and larger will have their own Test Plan. Small projects or one-off testing efforts generally fall under a Master Test Plan. Examples of small projects that are covered by a master test plan include banner and email testing.

An important point to take away is that all projects that need to be tested by the QA department are covered by at Test Plan of one kind or another – whether explicitly such as having its own Test Plan, or implicitly, such as an individual email that falls into a group of like testing deliverables. As of this writing, the Master Test Plan that covers all banner and email testing is the: QA Email and Banner Master Test Plan 3.0.

4.3.1 TEST PLAN PROCESS

The FCB Chicago Digital QA Department follows a process to create, review, and approve Test Plans for projects.

Step 1: QA Receives Deliverables
After receiving the relevant project receivables (BRD, Wire-Frames, Walkthroughs, etc…) the QA Manager can start creating the Test plan.

Step 2: QA Writes Features to be Tested section of Test Plan
Taking the information, the QA Manager writes the In Scope/Features to be Tested section of the Test Plan first. This section can also be referred to as the Decomposition of the requirements. Essentially requirements are broken down and outlined here. After this outline/decomposition is finished, the section sheds light on the whole picture of the testing effort. From here, everything else flows.

Step 3: QA Analyzes the Features to be Tested
Based on the now complete feature decomposition, any previous estimates can be updated; work can begin to be divided among testers, timelines can be adjusted (upward or downward) and new test cases could be started.

Step 4: QA Manager Completes the Test Plan
QA completes the rest of the Test Plan, preparing the document for review. See the Test Plan Structure section below for a review of all sections in the Test Plan.

Step 5: Test Plan is Reviewed and Approved
QA Manager presents the Test Plan to the key stakeholders of the project for review. After review and feedback, the Test Plan is approved.

4.3.2 TEST PLAN STRUCTURE

The Test Plan outline format we use here at FCB Chicago comes from the IEEE 829 format. From Wikipedia: the IEEE is the Institute of Electrical and Electronic Engineers. The IEEE is the world's largest association of technical professionals around the world. To paraphrase Wikipedia: IEEE 829 – 2008, also known as the 829 Standard for Software and System Test Documentation, is an IEEE standard that specifies the form of a set of documents for use in software testing and system testing. Here at FCB we tweaked the template to suit or project needs.

By leveraging the IEEE 829 format with our Test Plans, we can feel confident that our testing efforts will deliver quality products.

What follows is the outline of sections that are in every project Test Plan.

Test Plan Template

- Overview
- Features to be Tested
- Features not to be Tested
- Test Approach
- Exit / Acceptance Criteria
- Suspension / Resumption Criteria
- Test Deliverables
- Configurations to Test
- Responsibilities
- Schedule
- Testing Tasks

Overview – Summary of project and introduction of new features

Features to be Tested – Also known as "In Scope". This section includes the feature requirement decomposition. Every aspect of the project will be broken down into manageable testable sections to which to write test cases for.

Features not to be Tested – the former way of saying "Out of Scope". We're not testing the items listed here.

Test Approach – The QA team will be approaching testing the project in a certain way and that will be described here. Planning, Executing, Defect management, and Reporting will be spelled out so everyone will have the same expectations of testing across the project.

Exit / Acceptance Criteria – How does one know that they are done testing? Without this section, one doesn't really know when to stop – theoretically, one could test the same project for a year and still find new defects. Monetary and timing budgets often come into play here. One example of a criterion that FCB Chicago Digital leverages is "The project is not delivered with any Blocker or Critical priority defects.

Suspension / Resumption Criteria – These are the approved reasons why QA would stop and or resume testing code for a project. Common suspension criteria are the incorrect build is deployed to the staging servers, the site is having connection issues, PM's ask us to hold testing pending client approval, etc… Examples of resuming testing are when blocker defects have been resolved, connectivity has been restored, correct builds have been deployed, etc…

Test Deliverables – The items that the QA Department is responsible for delivering to the project team are listed here. These change often across projects and it is important to examine the list and make sure it's matching your expectations of the testing effort. Examples of test deliverables are a follows.

- Test Plan – This Document
- Test Case Report - Details of test cases are provided
- Test Run Report - Results from a test run are tabulated
- Defect Report - Defects entered on the project are listed along with their priorities.

Configurations to Test – Here are the Platforms, Browsers, and Devices that the QA team will be using during testing. This list changes often as well across clients and projects. This list must match what is typically provided on from the Statement of Work from the Project Management team.

Responsibilities – This is where the matrix of who is responsible for what task is spelled out. The rows on the left are not just the roles on the project, but the names of the actual individuals. The row along the top has the tasks that must be performed. This prevents risk in the project where individuals don't know who is doing what and tasks are missed. With the Responsibility grid, all critical tasks are assigned to individual team members and everyone is on the same page.

Schedule – The schedule of all testing related activities is here. Anything that is closely related to testing can be included on this table.

Testing Tasks – Tasks that need to be performed, Tools that are being leveraged in the testing effort and Types of testing including Functional, Smoke, Negative, etc… can be found here. Any miscellaneous testing items are here as well.

After the Test Plan is completed, reviewed and approved, if the test plan calls for them, the test cases now are created.

4.4 TEST CASES

A test case verifies that a specific piece of code is acting in the way that is expected. This can be a functional test including steps and data, or a non-functional test that simply verifies the correct language is displayed on a web page.

If a project needs test cases, QA will develop them in preparation of the testing effort. There is testing done without test cases; emails and banners are perfect examples. Emails and Banners testing use checklists to ensure that they pass testing. For bigger projects however, test cases are normally required.

A project is said to be 'tested' when all the planned test cases are executed across all planned environments. This is a typical exit / acceptance criteria as well. We'll discuss more of executing test cases in future sections of this document. For now, we'll discuss what test case elements are, and how a test case is developed.

4.4.1 TEST CASE STRUCTURE

As we said before, a test case determines whether a system or feature is behaving as expected. In order to achieve this, certain elements of a test case are needed.

- Title – Name the test case
- References – The name of the requirement or function that this test case is verifying
- Status – The status of the test case is here. Is it currently being worked on, new, or is it ready to run?
- Preconditions – This is the list of the things that have to be set-up before the test case can be executed. Perhaps the user needs to be logged in, or have certain redemption code handy to enter on step 5.
- Steps – The items the tester must execute in order to run the test case
- Expected Result – At the end of the test, what should the user expect to happen? At the end of a test case where the user intentionally enters a bad password at a login attempt, the expected result should be 'Wrong Password entered – try again' message is displayed.
- Other – There can be custom fields added to a test case as well, depending on the project and the test bed being used. Some examples include: Test Case Type, Section of site, and Priority of test case.

4.4.2 TEST CASE DEVELOPMENT PROCESS

QA starts creating test cases inside of TestRail. TestRail is the QA Department's new Test Case Management and Repository Tool. Tests can be created, reviewed, executed, and reported on all from TestRail.

Step 1. Review Receivables - After QA reviews the project receivables (wire-frames, BRD, etc.) and the requirements decomposition is created in the Features to be Tested section of the Test Plan, test cases can be started.

Step 2. Create Test Case – The QA team member writes test cases that verify the features to be tested. The test case has a status of "New" when the test case is first created. When the tester believes the test case is ready for review, they update the status to "Ready for Review"

Step 3. The designated test case reviewer reviews the test case marked "In Review". If the test case passes review, the reviewer changes the status to "Ready". If the test case needs some updates, the test case moves back to a "New" status.

Any New cases are being worked, any In Review are waiting to be reviewed, and any Ready cases are ready to be added to a Test Run and executed.

5 TESTING

This section, like some of the other here in this document, could be their own document by themselves. What follows is a detailed summary of how the QA Department tests digital projects here at FCB Chicago.

5.1 TESTING PROCESS

As in Section 3 of this document illustrates, regardless of the type of project under test, the overall testing process is the same: a Test Request is created, test planning is performed, approvals are received, and testing is started.

Testing is the QA Team-members examining a system to ensure that the requirements of the system are met. Any issues found are logged, tracked, and communicated to the stake-holders on the project.

The QA Department has processes put in place to test our three major types of projects: Banners, Emails, and Websites.

5.1.1 BANNER TESTING

FCB Chicago Digital Department creates hundreds of banner ads per year across almost all clients. There are three developers and at least two dev managers that cover the creation of these banners.

In addition to all the work that goes into these files, the current digital landscape is changing. A strong trend to move away from Flash banners and moving to HTML5 has started in recent months. Also, Google / DoubleClick are also stopping their testing of creative pieces, leaving the work to land on internal QA departments.

The QA Department follows the QA Email and Banner Master Test Plan 3.2 when planning for and testing banners. This Test Plan has been approved by the Project Management Team. Any changes to the Test Plan will need to be communicated to the PM's. Any major changes to the document should be approved by them as well.

The QA Email and Banner Master Test Plan 3.2 can be found in the directory included with this document. Please review this document for the current QA checklist for banners.

The PM attaches the banner files to the Jira ticket that is used to track the testing effort for the banners. After the PM receives their create approvals, they change the status of the ticket to "Ready for Testing". Changing the status of the ticket automatically creates a notification for the QA tester assigned to the ticket. At this point the tester has everything they need to test. Testing begins.

Requirements to start testing

1. Banner Testing Checklist
2. Banner files
3. A Green-light to start testing (PM clicking the 'Ready to Test' button in Jira)

Any potential issues found during testing will be entered into the Jira Issue tracking system and assigned to the designated individual that accepts new defects. Often times this is the developer of the banner. Sometimes the assignee is the PM, or the Dev Manager on the project. The ticket should specify who receives defects.

Once all the testing is complete, and any/all defects have been entered, testing is complete and the tester enters their comments/summary and changes the status on the ticket to "Tested – Cert Pending". When all tests pass, and any defects found have been resolved, then the status can be updated down the workflow to "Certified". Certified projects are ready to be deployed to the live environment. Refer to the Jira section of this document for detailed illustrations of the workflows of Test Request tickets.

5.1.2 EMAIL TESTING

Email (Campaign) testing is another large part of the FCB Digital's business. Many of our clients request that FCB Chicago QA perform testing on the email marketing campaigns every month.

Email testing has unique characteristics:

- Files are *Blasted* – There's no coming back after the files are sent out to potentially thousands or tens of thousands of individual inboxes. Unlike a website or a banner ad, if a blasted email has an error – development can't fix it on the fly. It's already out of here. Emails have to be correct the first time.
- Emails are probably the only type of project where QA focuses not only on the functional aspects of the item under test, but the creative too. QA is responsible for how the email displays across email clients and browsers. We have two email client testing systems that we leverage for this kind of testing: Litmus and Email on Acid. More to come on these below.
- Emails have legal requirements – Yes, there are End User License Agreements and Privacy Pages and Terms of Use screens with the sites that we develop and test, but emails must be compliant with the CAN-SPAM Act. The CAN-SPAM law spells out requirements that marketers must include in their email communications to the market.
  CAN-SPAM Act Compliance
  a. No False or misleading header information – The email must be relevant to the header info
  b. No deceptive subject lines – The subject line must mention what the email is about
  c. Identify message as an ad – Make sure the opener knows it's an ad
  d. Tell recipients where you're located – Where the office is located
  e. Tell recipients how to opt-out for future emails – Unsubscribe info is here
     i. QA not only verifies that the Unsubscribe page displays on clicking the Unsubscribe link in the email, but also verifies the Unsubscribe page functionality too. Testers are required to take a screenshot of the unsubscribe page while testing and attach it to the Jira ticket as a high level check that this has been performed.

For more information about CAN_SPAM, please refer to the following link:
https://www.ftc.gov/tips-advice/business-center/guidance/can-spam-act-compliance-guide-business

Like Banner testing, the document that covers email testing is QA Email and Banner Master Test Plan 3.2. The email checklist is included in the Test Plan.

Testing an email follows these steps:

1. PM requests testing and supplies creative / click map
2. Dev completes the email file
3. PM receives all necessary approvals for the email
4. PM gives QA the green-light to start testing

5. QA performs a test blast to Litmus, and any other email accounts necessary
6. QA Examines the email across email clients, completing the email checklist for each one
7. Any defects found are entered into Jira
8. After all testing is complete, the test request status is changed to "Tested – Cert Pending"
9. Dev updates file to fix any defects found
10. When any/all defects are resolved and all checklist items are checked, the status is changed to "Certified"

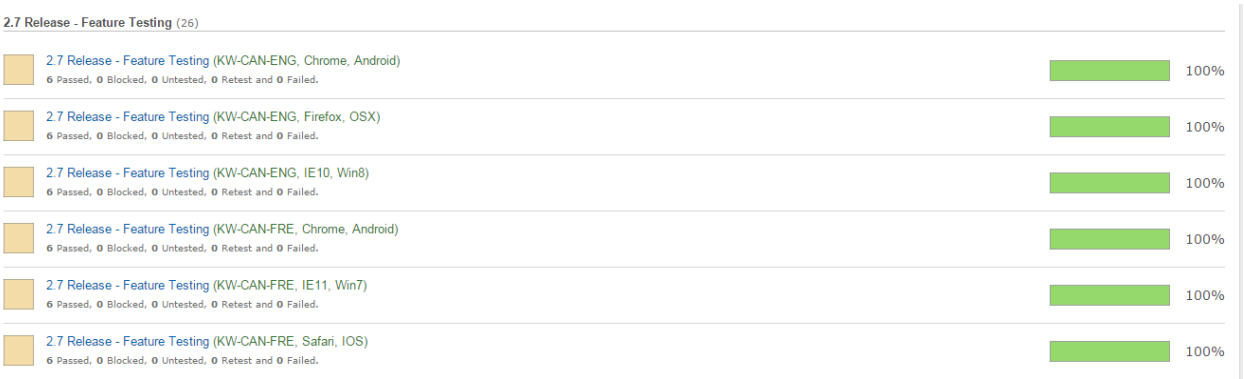
5.1.3 WEBSITE TESTING

Most of the web sites we build here at FCB Chicago are one to three page micro-sites. Some examples include a promotion for a product, a site to tell people about an event, and a site to share music videos or pictures of pets. There are some larger sites such as Aramark.com and Paccar's Parts and Service site as well.

QA is responsible for the functional quality across in scope browsers and platforms and devices of the sites that we create. The FCB Chicago QA process is mapped out is in section 3 of this document. From that diagram, we're in the Test Execution node here. Drilling down into the details is what's to follow.

If there were a heart of this document, this would be it. This is the section everyone thinks of first when they hear "QA" (although as the reader can attest to, there is far more to QA than just running tests.) As was mentioned earlier, the test case planning section has been (or should have been done by now) already performed. Planning in a way doesn't 'stop' in projects; it meshes with the testing phase by necessity. Plans change as unexpected requirements or environmental factors arise. QA must be able to have a flexible enough plan to pivot when necessary, managing risk along the way. Test execution is what the topic is here – but do not be surprised that planning often times comes up when discussing testing. They go hand in hand.

If it was decided in the planning of this project that test cases will be used for testing, then the testers will execute the tests at this stage. Test cases are created and updated in TestRail. Tests are divided out among Test Runs, and those are run according to plan. Projects can have hundreds of test cases or just a few per configuration. A configuration is a combination of environmental conditions that affect the test. Example: configuration 1 is Win 7, Chrome (latest). Another example is configuration 2: IOS v. 8.0, Safari (latest), iPhone 6. Tests can be run across all configurations or a section of them, according to the Test Plan.



Excerpt from Paccar's Parts and Service 2.7 Release – Test Run Progress Display – TestRail Test Case Management System

If no test cases are required or tests were required and they are completed with time to spare, ad-hoc / exploratory testing is performed. Exploratory testing doesn't use scripts or test cases; instead the tester tests the system combining three activities: learning, test design, and test execution. Testers don't have the structure of test cases, but are capable of finding issues because they are to learn how the system works as they are testing the system. A tester can approach a screen on a site and ask themselves, "What if I do this, this, and then this? Does that work, or do I see an error display on the screen?"

This methodology puts a lot of trust into the tester to find issues while constantly increasing the level of testing. As the tester tests the system, they are learning new things about the system and are able to fine tune their tests to uncover issues along the way.

Test case execution, by contrast, is more regimented. Test cases when written map back to specific requirements. The tester knows that a requirement is covered and works properly because all the tests that link back to the requirement have been run and all passed.

Is one better than the other? Not necessarily. QA uses both test case and exploratory methodologies to ensure that quality products are delivered. Both have their own advantages. Exploratory testing relies on the investigative approach of the tester who could uncover issues that were not planned for. Test Case / Script testing will ensure that all requirements that need to be tested are indeed tested, thus ensuring that all requirements are verified. There is an investment of time at the beginning of the process in order to write the test cases. Many projects that have repeatable components can benefit greatly from test case testing as the tests, like the code, are being re-used.

Unlike Banner or Email testing, Website testing doesn't have a checklist per se. The decision on whether the product is ready for deployment to production can be determined by the Exit/Acceptance Criteria, mentioned in the Test Planning section of this document. Once these items are all met, the product can be delivered.

5.2 TESTING TOOLS

The Quality Assurance Department leverages the following tools during the QA Process.

5.2.1 TESTING TOOL: JIRA

QA tests and when they find issues, they are entered into the Atlassian System called Jira. Jira is very robust platform and our instance is cloud-based. Any user of the system can access Jira from anywhere. The system is easily configurable allowing users to create not just defect issues, but testing requests, features, tasks, and feature requests. Each one of these issue types has their own configurable workflows that guide the issue from beginning to end – ensuring that the needed resources are working on the issue.

What follows in the Jira section of this document is a high level summary of Jira, what ticket types are available, and how a ticket goes through a workflow.

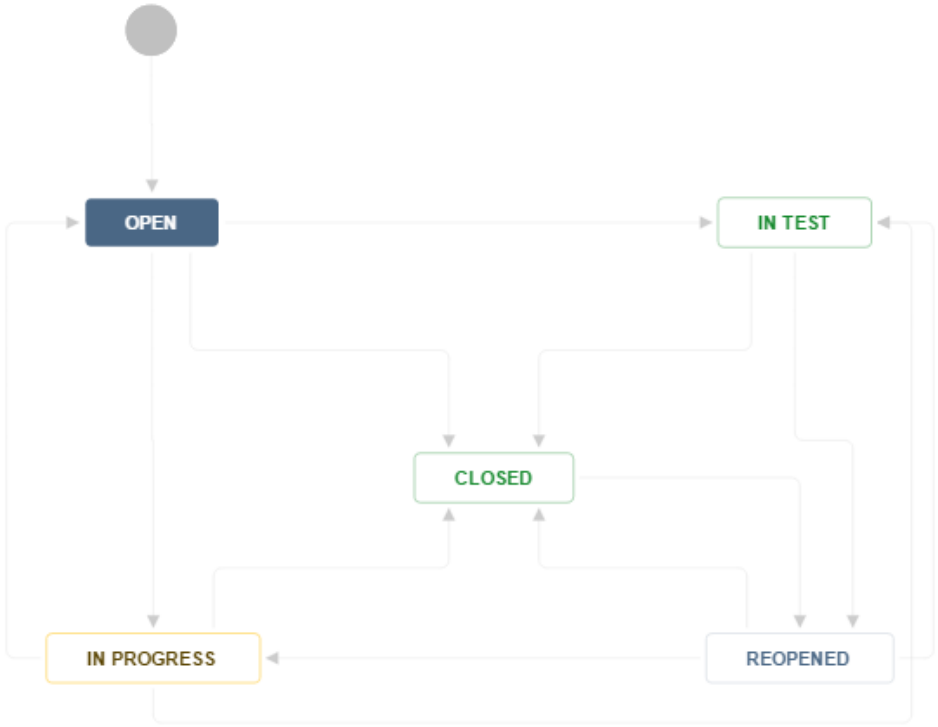
There are default issue types in Jira, and the admin can also configure their own custom issue types too. The out of the box types we use are Defect, Task, Sub-Task, Feature, and Feature Request. Each one of these issue types is a collection of different information points that is logically related to the type itself.

For example, a Defect issue type will have an Expected Result field, because if the tester is logging defect, they are seeing the system behave differently than was expected. This field though is not necessary in a Test Request.

5.2.1.1 ISSUE TYPE: DEFECT

This is the issue type everyone thinks of first. This issue type contains all the required fields to report/track a defect with the system under test. Fields include: Expected and Actual Results, Fix Version, and Description. The tester entering the defect also enters the steps on how to replicate the defect here.

Defects use the Default issue workflow in Jira in almost all projects (there is one that has a slightly modified version of the default version).



Defect / Default Issue Workflow in Jira

Defect Issue Type Scenarios

Scenario 1*: QA logs an issue into Jira. The ticket has all the relevant information that the team needs to triage and deal with the issue. In the simple process flow, the Development Manager receives the defect, assigns the ticket to a developer who fixes it, marks the defect as Fixed and assigns it back to QA for retesting (verification that the defect is fixed). QA verifies the fix, and closes the issue.*

Scenario 2: *QA logs an issue, Development Manager determines that the issue is not valid and closes it with a resolution of "Won't fix" and adds a comment as example 'working as designed'.*

Scenario 3: *QA logs an issue; the Development Manager assigns the issue to a developer who addresses the issue. The developer marks the issue fixed, assigns it back to QA and the tester retests it and finds that the issue remains. The tester adds their comments and changes the status to 'Reopen' and assigns it back to the developer.*

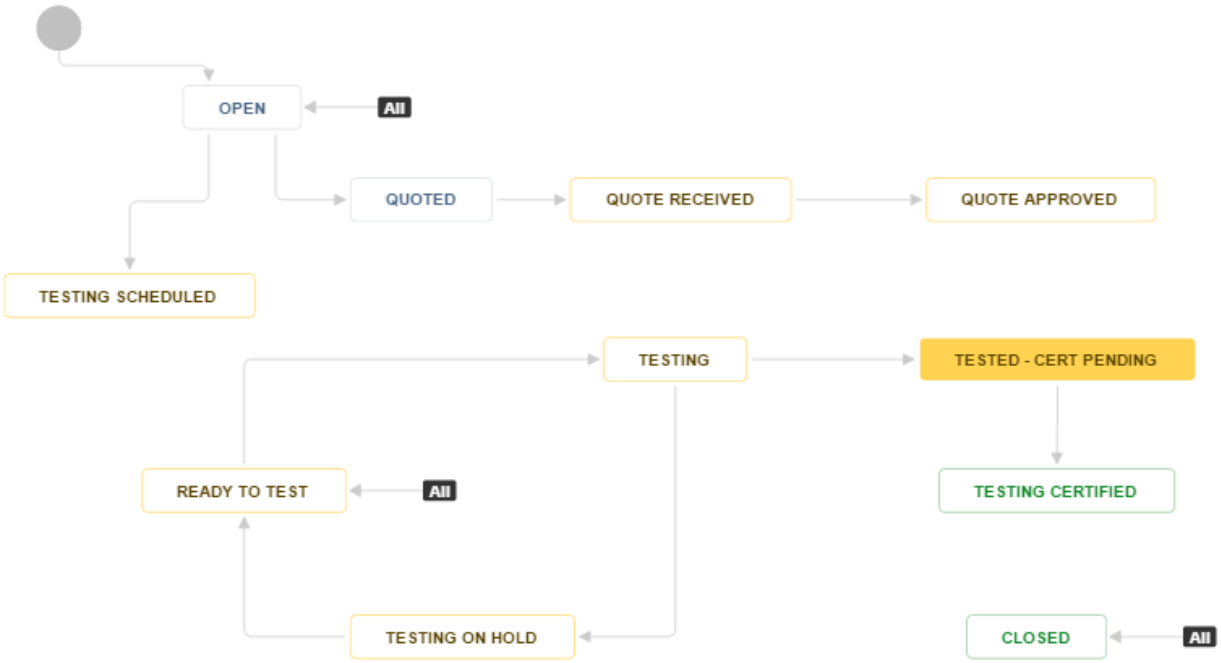
5.2.1.2 ISSUE TYPE: TEST REQUEST

I have created a custom 'Test Request' issue type to track testing across projects. The need for a test tracking issue type became very apparent when PM's would ask me to test different projects across different accounts with different documentation platforms being used (Basecamp being an example of one document platform). I needed to house and track all testing in one place – and with one type of ticket.

A test request ticket is created in almost all cases by the Project Manager. The PM enters the required information and assigns it over to QA. When QA receives the ticket, the QA Manager starts preparing for testing (see section 4. Test Planning).

The Test Request has all the required fields for planning testing for a project.
Job Code, Platform and Browser to test on, key dates for testing, testing link, attached project receivable documents such as link matrix, business requirements document, etc..

When the Test Request is created, it will enter the workflow below as an "Open" ticket. See below for the Test Request workflow scenarios.



Test Request Issue Type Workflow in Jira

Test Request Issue Type Scenarios

***Scenario 1**: Normal Flow*

PM creates a Test Request and the QA Manager receives it. The QA Manager inspects the details provided and starts test planning. The request is assigned to an in-house QA tester, and the ticket is updated to "Scheduled".

When the system is ready to test, the PM updates the ticket status to "Ready to Test". QA receives an automatic notification that the system is ready for testing. QA starts testing and updates the status to "Testing". All watchers on the ticket receive a notification that testing has started. Issues are found and logged, testing is completed, and the tester changes the status to "Tested – Cert Pending".

***Scenario 2:** Testing is put on Hold*

Scenario starts as above, but during testing one of the criteria listed on the Suspension Criteria list of the Test Plan occurs and testing stops. Testing resumes after the criterion are met on the Test Resumption list of the Test Plan.

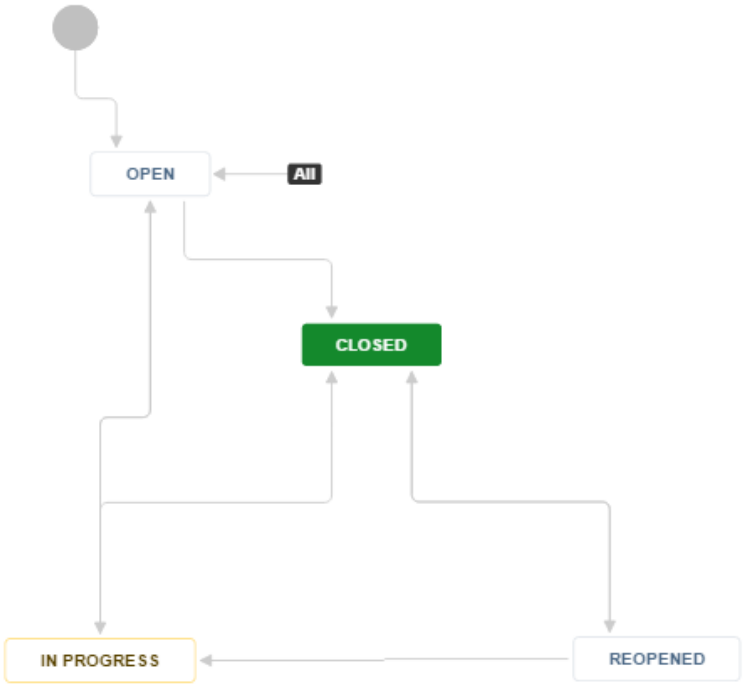
***Scenario 3:** Work needs to be tested out of house*

PM creates a test request. The QA Manager finds it necessary due to resource/time constraints to ask one of our testing vendors to test the system. The ticket is assigned to the QA Vendor, and status is updated to "Quoted". The vendor replies with a quote (outside of Jira – by email), ticket changes status to "Quote Received". The quote is next approved by the PM. QA Manager updates ticket to "Quote Approved". The flow now transitions into the normal flow above when the project is ready for testing.

5.2.1.3 ISSUE TYPE: TASK

This is a smaller issue type with just basic fields used to track tasks that need to be performed. These tasks are not testing related. Examples include, deploying code at a certain time, reviewing code, etc…

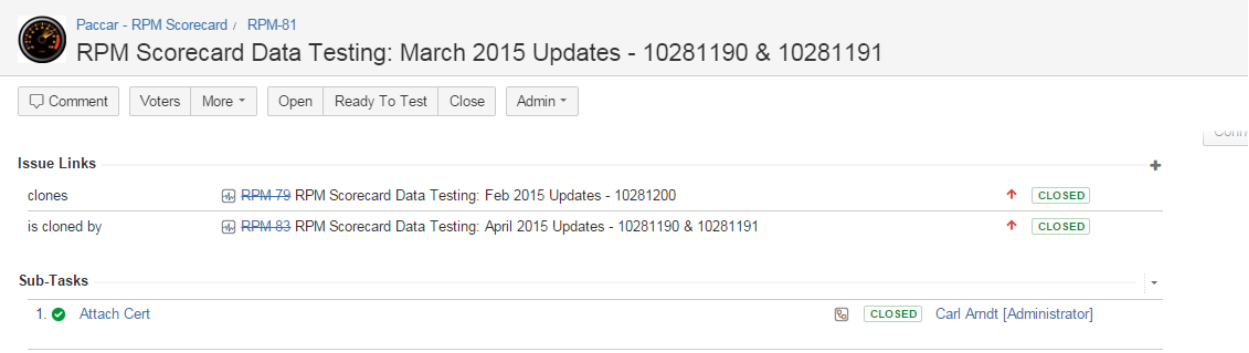
Task workflows are also simple.



Task workflow in Jira

5.2.1.4 ISSUE TYPE: SUB TASK

This task is identical to the Task with one difference: it is a task that is included within another ticket. A Test Request ticket might have the following subtasks: Create Test Plan, Create Test Cases, and Attach Certification. Subtasks must be completed before the holding ticket can be completed.



Excerpt of ticket that includes a sub task ("Attach Cert").

5.2.1.5 ISSUE TYPE: FEATURE

A Feature ticket is an issue that describes a change or update to the system that needs to be developed and tested. Any planned work can be tracked this way. Currently the basic fields are included in a feature issue.

5.2.1.6 ISSUE TYPE: FEATURE REQUEST

This issue type probably gets more activity than the Feature issue type actually. Here's the issue type you can have all users on the Jira system who are testing the project's website enter their requests on what the site should do, how it should behave, etc… If there wasn't a Feature Request type, users would be entering tasks or defects (or sending lists in an email) to get their thoughts expressed.

The Feature Request solves a few problems:

- They keep non-issues out of the defect issue type tickets
- Feature Requests allow users to express how they want the site to work
- Feature Requests will allow PM's and stakeholders to track and decide on whether to add these features now, or in a future build, or not at all.

5.2.1.7 OTHER ISSUE TYPES

Jira allows the administrator to create more issue types as the need arises. There are also some other issue types included out-of-the-box with Jira that are disabled in our instance here at FCB Chicago that can be activated any time based on need.

5.2.2 TESTING TOOL: TESTRAIL

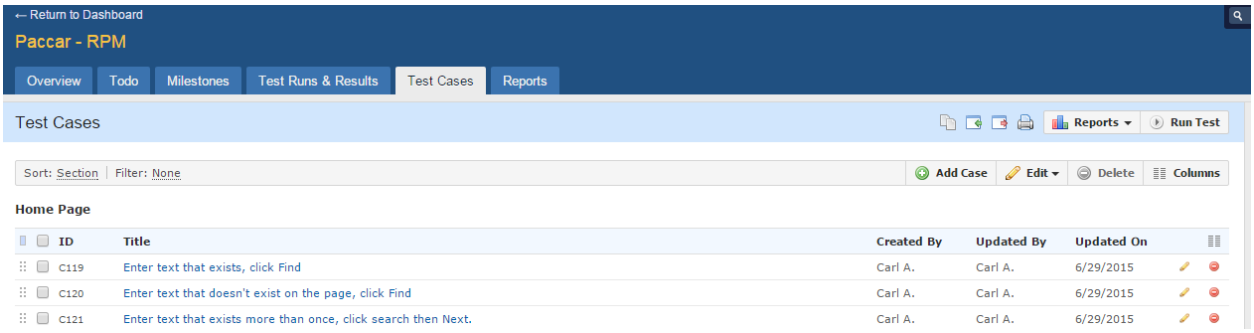
After the Test Plan is created for a project and it is decided that test cases are needed, tests are next created here. Test Rail is a test repository across all projects and accounts. With this relatively new tool for FCB Chicago, all QA team members can create, update, run, and report on tests. For non-users of the TestRail platform, reports can be created that generate automatically and notify anyone that needs them.

5.2.2.1 TEST CASE PLANNING

Planning Test Cases comes in two parts: Test Cases, and Test Case Runs. Test Cases can be created and saved over time and reused by one or multiple Test Runs.

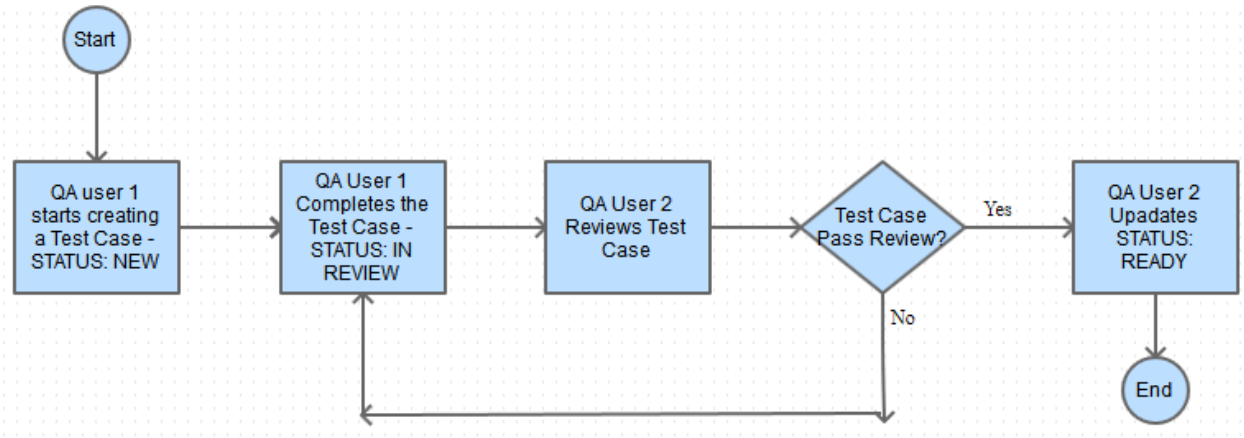
5.2.2.1.1 TEST CASES

Test Cases are located in the select project's Test Cases tab. Each TestRail project has this structure. Tests are created here and can be organized by sections. In the below example, the Test Cases are organized by sections based on pages of the site starting with the 'Home Page' section.



Test Cases tab in TestRail

Test Cases go through a small workflow when they are being created. The review process ensures that the tests are worked by at least two QA individuals. One person creates the test, and one person reviews the test. Tests created this way should be more robust, better detailed, and may also provide better test coverage.



Test Case Creation Workflow - TestRail

5.2.2.1.2 TEST CASE RUNS

When the test cases are created, and they're ready for the Test Run. A Test Run is a collection of test cases that are typically run together. They needn't be run in sequence (as a well written test case is autonomous and should have no dependencies outside of itself). A Test Run can be organized by date, or by a select set of features per project, or any other method that makes sense for your project. Test Runs can be copied and used again, but in TestRail you can't edit the

name of copied Test Runs so be careful how you name it if you're planning on using it again in the future.

5.2.2.2 TEST RUN PROCESS

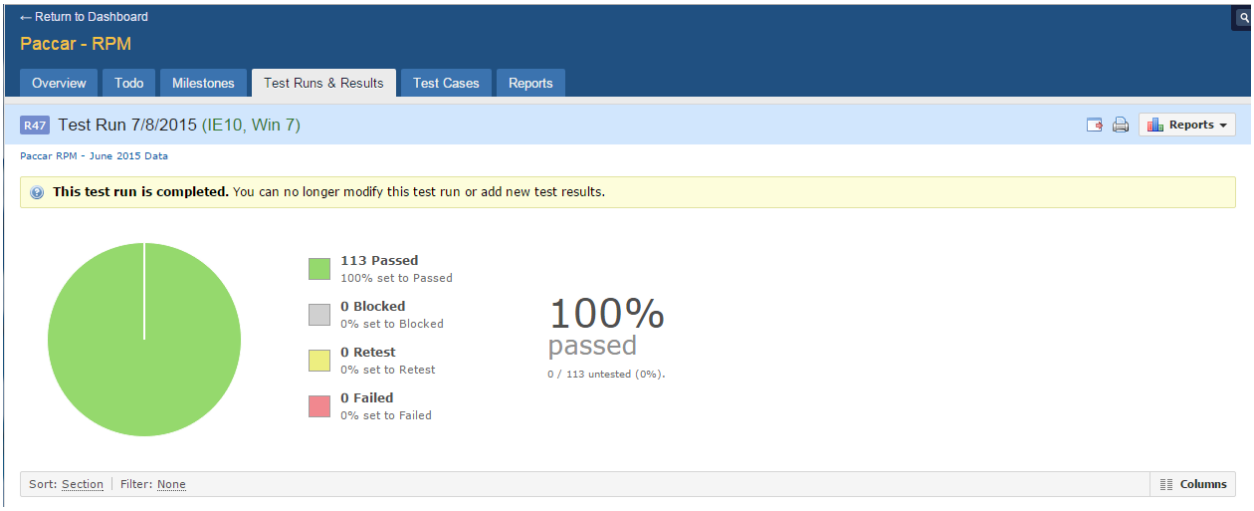
Test Runs are executed by selecting the project to test, clicking the Test Runs & Results tab, choosing a Test Run and then a Test Case. Then they follow these steps.

1. Click "Start Progress"
2. Run the Test Case
3. Click "Stop" (or "Pause" and then "Resume" if needed)
4. Choose a status for the test case executed: Passed / Blocked / Retest / Failed
5. Enter a comment. This comment must include a defect id for any Blocked or Failed
6. Add any attachments if needed
7. Click "Add Test Result"

Upon completion of the above steps, the Test Case inside the Test Run will have the new chosen status; the result will be tabulated and be reflected in the Test Reports.

5.2.2.3 TEST REPORTING

The TestRail platform has many options on configuring reports. Perhaps the simplest report is just the Test Run Summary page. It provides many critical KPI's without creating a report: How many Passed, Failed, Blocked, and Retests we have. Displays a pie-chart summary and displays the percentage completion.

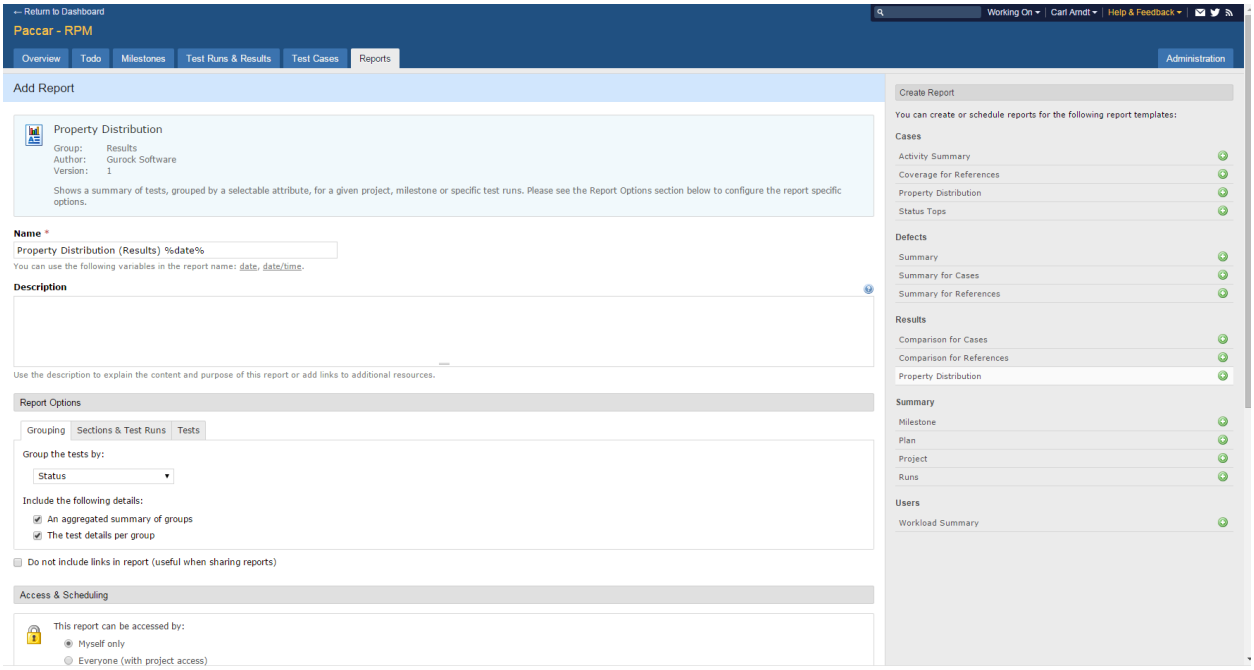


Test Run Summary Screen – TestRail

A different report I use is the "Property Distribution Report" – that's what TestRail calls it, but you can change the name of the report when it runs. To access the reports, navigate to the Project, then the Test Reports tab. Click any one of the green plus buttons to the right of the desired report. I have included a screenshot of the New Report Screen below.

The interface is simple enough to navigate and create the report. One thing I found key here is a report can be configured to run now, or a report can be scheduled to run. A scheduled report will automatically run and email the desired team members the report. There are two formats to these reports: one for users of TestRail (just QA team members) and one for non-users. Ensure that when you enter non-user emails into the test screen that they receive the non-user formatted report email. The email will contain the HTML for the page so they simply have to open the file to see everything – no login required.

If any team member require more KPI's in their reports QA can configure the report to provide them.



Report Create Screen – TestRail

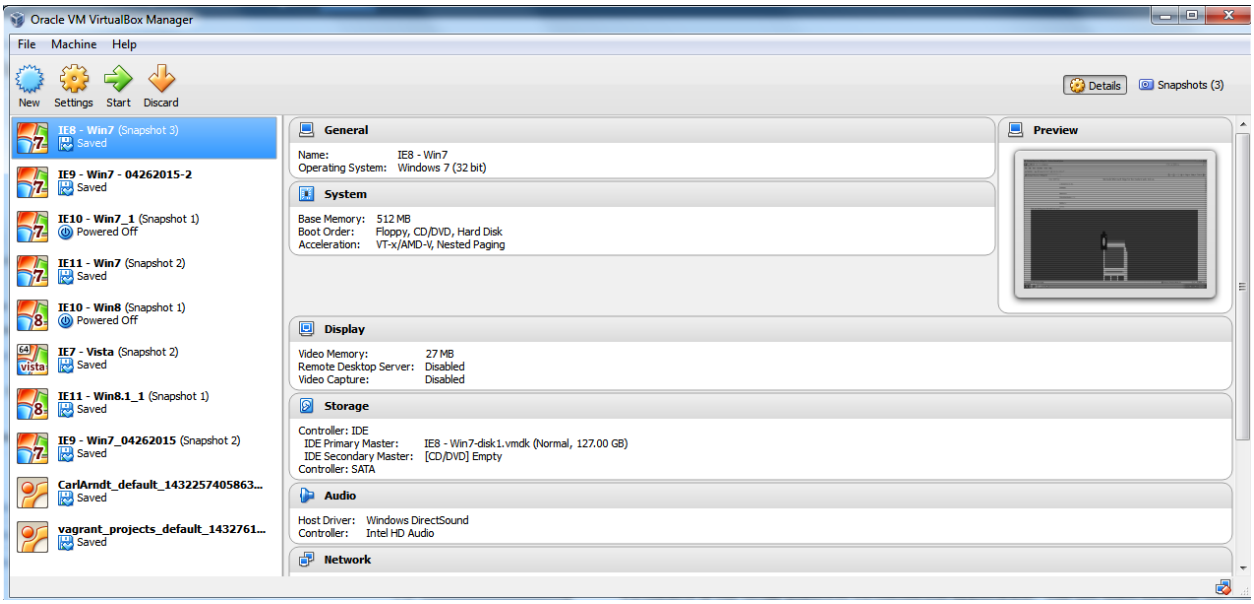
5.2.3 TESTING TOOL: VIRTUALBOX

VirtualBox is our cross-platform virtualization application. The QA team extends the capabilities of our assigned laptops to run multiple operating systems at the same time. When QA is required to test a website across different platforms and browsers, VirtualBox is our first tool we leverage.

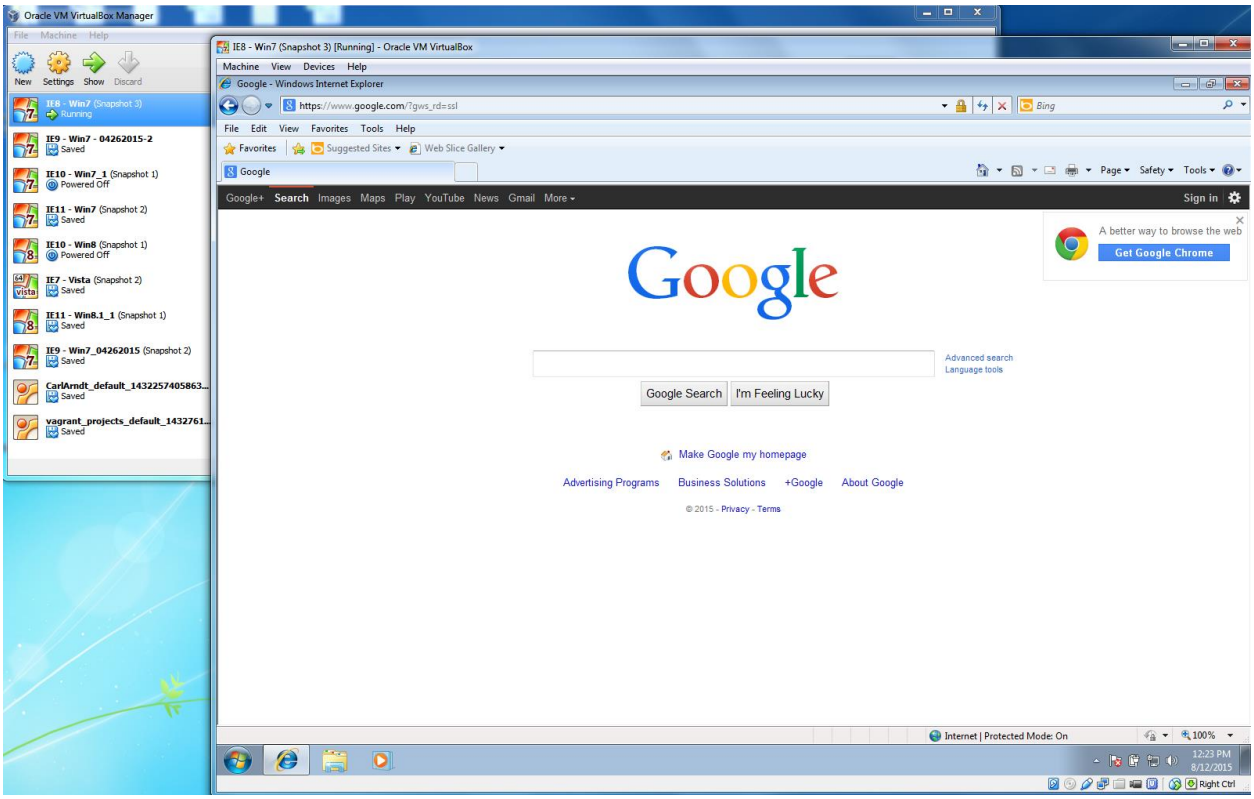
The VirtualBox application is free, and the virtual machines that must be downloaded and configured on VirtualBox are also free. All windows OS's are available (no Mac's yet and I don't anticipate them to ever be available as a VM.)

Below you will find a screenshot of the Oracle VirtualBox Manager. The manager window controls the virtual machines once you have installed them. Whenever a tester needs to test on a certain platform, they select the platform on the left side of the window by double-clicking. The platform window then loads and displays in a new window on the tester's machine.

QA is responsible for keeping the VM's up to date with the required versions of the browsers and platforms needed for testing.



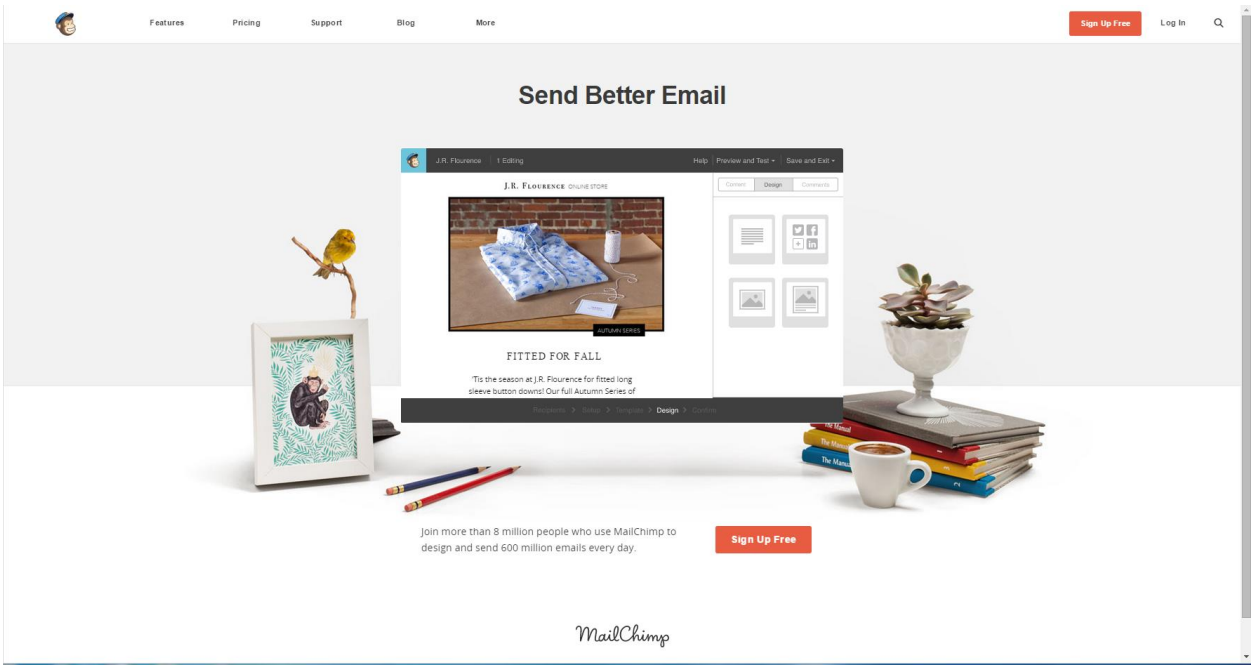
Oracle's VirtualBox Machine Window



VirtualBox Machine – Win 7 / IE8 Virtual Machine Running

5.2.4 TESTING TOOL: MAILCHIMP

Mailchimp is the tool that sends targeted campaigns and marketing emails. FCB Chicago QA uses Mailchimp for testing emails by sending test blasts to ourselves and to Litmus.



Mailchimp Home Page

FCB Chicago has multiple accounts to access Mailchimp. Refer to Browserstack for credentials or contact any of the Digital team members for help.

Sending an email blast from Mailchimp follows these steps:

1. Open a browser tab and navigate to Litmus
2. In Litmus, click 'Create a New Test'.
3. Select the desired email clients and submit.
4. A window with a test email address displays. Copy this email address – you'll need it in Mailchimp.
5. Open a new browser tab and navigate to the Mailchimp site.
6. In Mailchimp, click the Lists tab in the header. The Lists page loads and displays.
7. Choose a current list or create a new one to blast the email to.
8. Follow the steps to add a subscriber to the list and add the email address you received from Litmus. Save the List.
9. Click the Campaigns tab in the header
10. Create a new Campaign and name it with the file name of the email you are going to test.
11. Choose 'Regular Campaign' and click Next in the lower right corner.
12. Choose your list you added the test address to, click Next.
13. Setup Page – Enter the file name under test in the Name, Subject, and From fields, click Next.
14. Template Page – Choose 'Code Your Own', then select the email zip file delivered to us from Development. Upload it.
15. Design Page – When the file is uploaded, the Design section is selected and the email preview window displays. Click Next.
16. Confirm Page – Click Send when ready.

17. Click back over to the tab with Litmus and click 'I've sent it'.

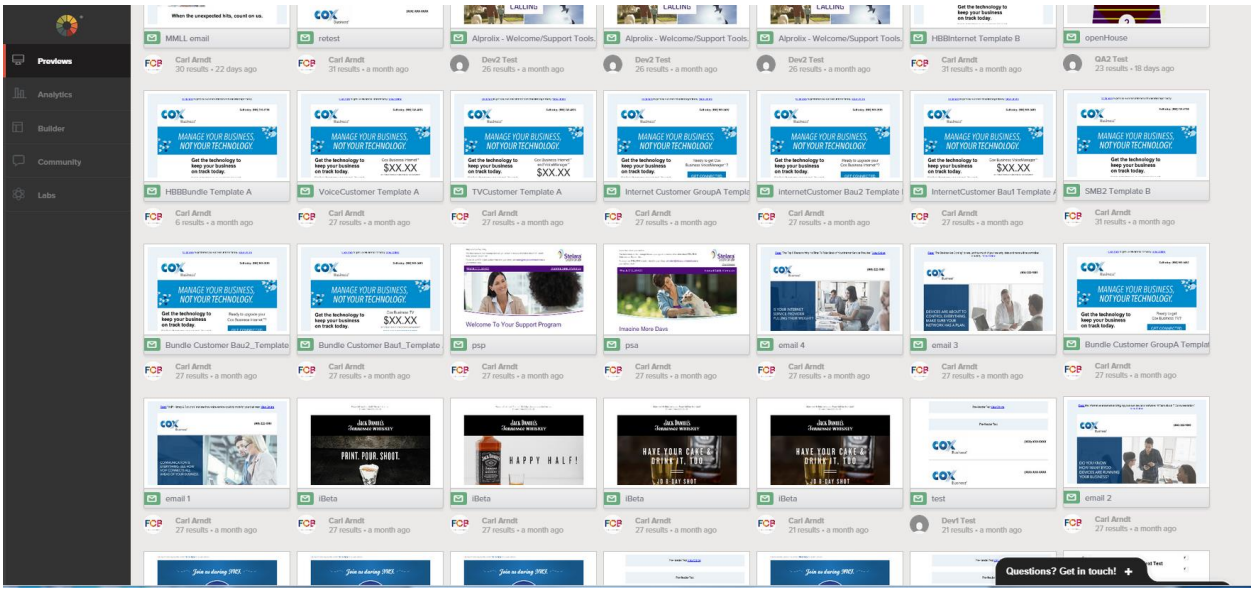
On send, the Mailchimp platform sends out the email to all addresses on the recipient list. The next steps are in Litmus.

Helpful Mailchimp Hints:

- When removing email addresses from lists, delete them, do NOT unsubscribe them. If you unsubscribe them you will not be able to add that email address to the list without a many-step re-subscribing process (due to Mailchimp's built-in CAN-SPAM Act compliance protections.)
- When entering the email subject, use the email file name. It's much easier to see which email is which when sending multiple test emails to the same inbox.
- The From email address must be a properly formed email address

5.2.5 TESTING TOOL: LITMUS

To test efficiently across many email clients FCB Chicago's tool of choice is Litmus. Litmus allows the tester to blast a test email to its configurable email client list so the email can be viewed many clients. Email code maybe the same for a campaign, but each email client interprets the code a little differently so verifying they are displaying as expected across clients is a must.



Litmus Preview Screen

Steps to Test in Litmus

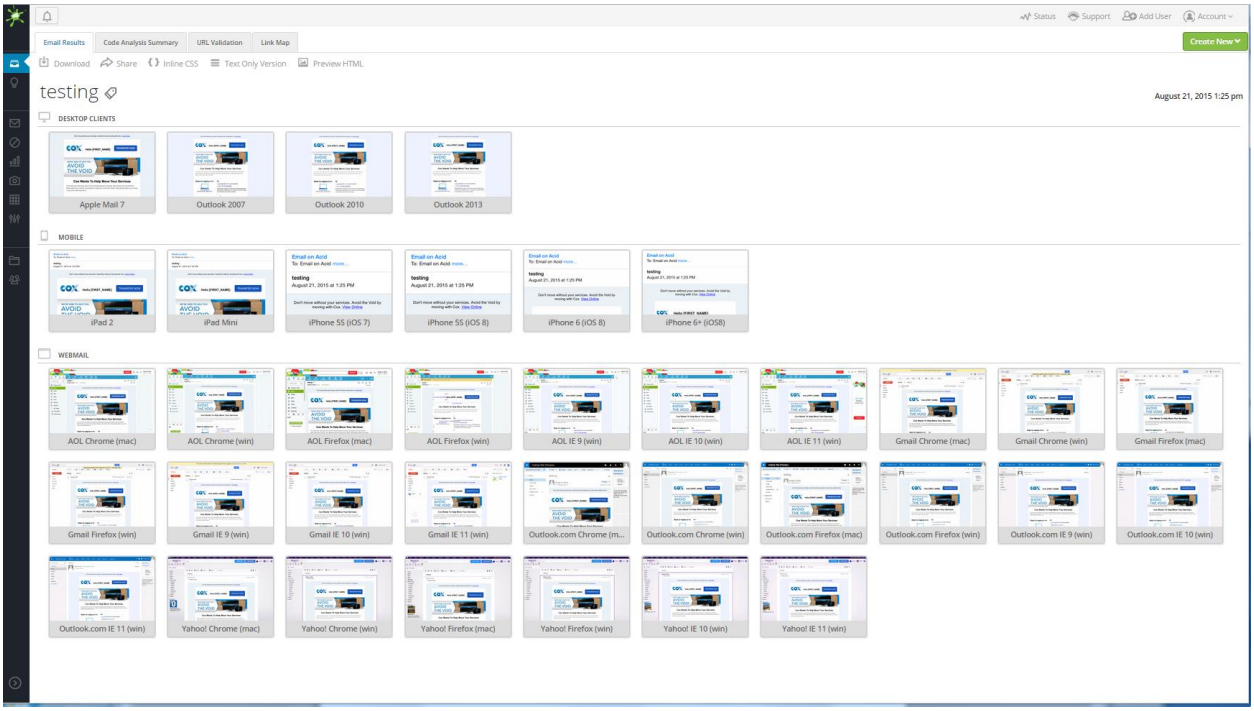
1. Log in to system and go to the Preview screen
2. Click the 'New Email Test' button
3. Observe the Email Testing Options screen loads and displays
4. Choose the desired configurations that will be receiving your test email
5. Click 'Start Test' at the bottom of the page

6. A 'Test Your Email' window displays next with an email address
7. Copy the email address; this is where you need to blast the email too from MailChimp
8. After blasting the email in MailChimp, go back to Litmus' Preview Screen
9. Click the latest email displayed in the Preview list which will be the email just received
10. The screen that loads next is the Email Screenshots page
11. Go through each one and verify that the email is displaying as expected

Note: The Litmus Screenshots page is just a list of screenshots, so no links will be functional in the test email here. If the email links need to be tested the email will need to be sent to the tester's email address so they can open the email normally.

5.2.6 TESTING TOOL: EMAIL ON ACID

FCB Chicago's second Email client test tool is named "Email on Acid". This tool works just like Litmus in that it receives emails, distributes them across email clients, it has a summary screen, and has a detail screen for each email. EOA however, has a shortcut of sorts over Litmus – the email doesn't need to be 'blasted' to EOA like with Litmus. You can instead just import the zip file to the platform and you're ready to start testing.



Email on Acid Summary Screen

A critical benefit of using EOA along with Litmus is the validity factor. A perceived defect found in Litmus can be validated and replicated on EOA. Once the issue is replicated on the second

platform, the issue ticket holds more credibility that it is a real issue, and not just a glitch on the Litmus system.

6 REPORTING

The Quality Assurance team reports on testing preparation, test execution, and test completion, in various appropriate levels and metrics. This section will describe the reporting process and what is delivered to whom.

6.1 REPORTING PROCESS

As discussed above, there are different projects tested here at FCB. The larger projects that need Test Plans are typically where most of reporting originates from. Before we get into the methods of how information is sent, the overall approach is updates to all stakeholders on the project at least once per day.

Jira ticket – A test request ticket is created and all stakeholders are added as watchers. Any update to the ticket generates an automated email notification to all watchers on the ticket. When the QA Tester enters a comment on the project that testing is 50% complete, or that he or she is stuck on an issue, all needed team members automatically know about it.

Team members can add themselves as a 'watcher' to the test request ticket. Watchers will receive the auto-notifications on all test requests. From a reporting perspective, these notifications serve as a reporting tool to enhance the communication process between the QA team and all stakeholders on the project.

What follows in the next section is a description of the different formats that QA communicates to the team.

6.2 QUALITY DELIVERABLES

6.2.1 DEFECTS

Defects are the most basic, and critical, form of reporting that the QA team has at their disposal. Defects tell the story of what is expected to happen with a system, and what is actually happening. Platform/Browser/Device, steps, description, versions of software, and screenshots are entered by the tester to make the development team aware that a fix is possibly needed.

6.2.2 TESTING SUMMARY REPORT (EOD REPORT)

What was tested? Where was it tested? How much more testing is needed before we can say 'we have tested everything'? These questions are answered in the Testing Summary Report.

Although the report has a few names, End of Day report, Summary report, QA Status report, etc… basically the same information will be included:

Project Name: <Project Name>
Job code: <job code>
Jira Ticket #: <Jira test request number>

Devices / OS:

OS	Device/Browser	Version

Defects Blocking Testing:
<Blocker Defects listed here >

Testing Activities:

- Executed test cases <Details>
- Testing is x% complete
- Hours used / Hours Total Remaining

Comments / Questions / Concerns:

- <comments>
- <questions>
- <concerns>

Defects Logged:

Displaying **3** issues at **02/Sep/14 5:31 PM**.

Issue Type	Priority	Key	Status	Summary
Defect	Critical	ABC-001	Open	<summary text of defect>
Defect	Major	ABC-002	In Test	<summary text of defect>
Defect	Minor	ABC-003	Closed	<summary text of defect>

Generated at Tue Sep 02 17:31:15 CDT 2014 by Carl Arndt [Administrator] using JIRA 6.4-OD-04-006#64001-sha1:b1633a7e36a916f95ac968aafe0f8b849ce804e3.

Total Defect Count for Project:

Blocker	0
Critical	1
Major	1
Minor	1
Trivial	0
Total	**3**

QA Test Summary / EOD Report

6.2.3 METRICS

For projects where it is required to report on metrics, QA will include these KPI's in the summary as well.

- Tests Executed / Percentage of tests executed
- Tests Passed / Percentage of tests passed
- Tests Failed / Percentage of tests failed
- Defects found
- Defects by Status
- Defects by Priority
- Defects by Assignee
- Hours used testing / Hours remaining to use on testing

6.2.4 QA CERTIFICATION

When the project meets all the criteria of the Acceptance/Exit Criteria list in the Test Plan, the QA team 'certifies' the project. Once a project is certified, the Project Manager (or Release Manager) can be sure that the project under test can be released to the client. Typical projects that have been certified have all major and up priority defects reported and resolved, the build has been tested across all supported browsers and platforms/devices, and all requirements have been met and are functioning as expected.

QA Certifications were implemented so that PM's know when everything is done with QA. If we didn't have a QA Cert document, the question of "Is QA Done?" can almost always be asked (and never answered satisfactorily).
"QA has made their first pass and has logged 3 defects. Are they done?" – No
"QA has completed three rounds of testing on the email files and 1 minor defect is unresolved. Is QA Done?" – No
"QA spent 1 hour ensuring that the latest HTML5 replacement banner on the Double Click system has been inspected according to the agreed upon FCB Digital Banner Checklist #3 and no issues were found. Is QA Done?" – If QA adds the Certification, yes, we're done.

And this brings up a very important point. The word "Done" can have many meanings, but in the QA world – "Done" means that all Exit/Acceptance criteria have been met. I needed to be very specific on how we're defining this word so there is no ambiguity. We can't have builds going to production without being done.

For more information on the Exit/Acceptance Criteria, see the Test Planning section of this document (Section 4.3.2).

FCB Chicago QA
HTML5 Banner Certification
<Project Name – Job Code>

Created by: Carl Arndt, Quality Assurance Manager

Summary

HTML Banners Checklist (List 1)

☐	Browser Compatibility: IE10
☐	Browser Compatibility: IE11
☐	Browser Compatibility: Microsoft Edge
☐	Browser Compatibility: Firefox
☐	Browser Compatibility: Chrome
☐	Browser Compatibility: Safari
☐	ClickTag - Banner Exit Click / ClickTag Check
☐	Other Links - Verify links
☐	Format - Verify Formatting / Images
☐	Animation - Verify animation is smooth and speed is appropriate
☐	Animation - Verify the non-user-initiated animations do not exceed 15 seconds (Initial load)
☐	File Size - Verify the size of the initial or primary file doesn't exceed spec (150kb guideline)

** Not all Banners are clickable*

Testing Details

QA Resource Certifying	
Date(s) of Testing	
Product / Build Version	
Date of Certification	

Release Checklist

Complete?	Task Name	Date Completed	Completed by
☐	Test Planning Complete		
☐	Test Case Execution Complete		
☐	Any / All Defects Resolved		
☐	Exit/Acceptance Criteria met		

Certification

The build has PASSED Testing and the Exit / Acceptance Criteria

The person signing below certifies that all the Release Checklist items have been completed and Testing is Complete.

Digital Signature:

QA HTML5 Banner Ad Certification Template