**QA Guidelines**

**Receiving a task**

Tasks will be received as an email or note in Attask. Select ‘Work on it’ to switch it to your active list. Once you have access to the project, download and view all related documents (PDFs, PSDs, and XLS). Once you have downloaded and reviewed the documents you should respond to the PM and ask any questions that aren’t answered in the docs.

**Test Cases**

With documents in hand and post review of the documentation you can begin writing test cases. Test cases are a list of every testing scenario that should be run through in a round of end to end testing. There is a test case template to follow. Test cases are divided into 4 different tabs: Functional, Creative, Analytics, and Performance.

**Functional** test cases will cover a majority of your testing. Everything from hyperlinks to data captures will be in the functional section. You must think outside of the box and think of creative ways to break the site. Be as descriptive as possible because this will be shared internally and with the client.

**Creative** test cases will cover images and copy. This should be a very simple, general description of how to match the comps to the test site.

**Analytics/Tracking** will be based on the requirements documents sent over by the PM. These test cases are just to verify that the correct tracking information is in place.

**Performance** testing includes using simple tools that grade a website based on the structure of the code and response time. Tools such as *Yslow* and *Pagespeed* are used in performance testing.

The Functional and Creative test case tabs of the Excel spreadsheet will be split into different columns: Test Case ID, Description, Objective, Pre-Condition, Action, and Expected Result.

**Test Case ID** is a unique number assigned to each line of the test cases. Ex. 1.00, 1.01, 1.02, etc.

**Description** is a quick and short write-up of what will be tested.

**Objective** is what the goal of the current test is.

**Pre-Condition** is any condition that needs to be met before the line can be tested.

**Action** is what needs to be done to complete the test—actual steps taken.

**Expected Result** is what *should* happen at the end of the test.

The test case line items of the Functional and Creative test case tabs of the Excel spreadsheet usually follow this format below, but they can vary at times according to the specific project being tested.

**Miscellaneous items** – could be one-off tests: Ex. an initial splash page to enter the site when first visiting the site. Only deleting cookies will show this splash again upon visiting the site.

**Global elements –** sameitems that appear on all pages throughout the entire website.

**Age Gate testing** (if applicable to the project) – All visitors to the site/page need to be of a specific age or older in order to enter the site—this is a legal requirement. Ex. separate test cases for testing the day before, the day of, and the day after the supported age given to us by the client. Same test cases follow for the month and year.

**Cosmetics –** includes compatibility testing on all supported browsers, mobiles, and tablets.

**Functionality –** we refer to site wire frame(s) for functional test cases.

For each/every page in the website being tested we create separate cosmetic test cases, functionality test cases, and tracking test cases on their respective and separate spreadsheet tabs.

Refer to the test case template to get a feel for all the items discussed in this test case section of this guide. Once test cases are completed they should be sent to the QA Lead and the PM of the project to review and give feedback on. Once finalized, they should be uploaded to Attask and/or SVN.

**Attask**

This is the main defect tracker used by Studio projects. If you are asked to email issues or write them out on a spreadsheet, speak to your QA Lead.

Once you begin testing and are ready to enter issues, open Attask and find the project you are working on. Select ‘New Issue’ and follow the format for entering new issues:

**Title** – [Page] ‘Browser if specific’ – ‘Issue Title’

**Description** – ‘Screenshot’ ‘Browser if specific’ - List steps by numbered order to recreate issue if it’s a complex issue or a detailed description for more minor issues.

**Assignee** – Developer or Project Manager (depends on who takes responsibility for the issue)

**Status** – *New* – This is the default for new issues to the Developer, or *Awaiting Feedback* – Default for new issues to Project Manager. Once the issue is reassigned back to us for testing then depending on the result(s) of the testing the status will move to one of the following: *Reopen*, *Closed*, *Passed in test*, *Passed in UAT*, or *On hold*.

**Defect Life Cycle:**

1. Issue reported (new)
2. Assigned to developer (open)
3. Fixed and developer sets as “Retest”
4. QA retests the issues
5. If item is fixed we “close” it; if item is not fixed we “open” it again and reassign it

**Issue** **Priority Levels** –

**Priority 1**: Functionally is broken, which prevents further testing. Examples: sign up page doesn’t load or submit button when clicked breaks the page. Also anything legal (like an age gate, compliance issue, terms and conditions) will always be P1.

**Priority 2**: Functionality works, but not as intended / Or a major creative issue. Example: the sign up page loads the wrong page or broken page.

**Priority 3**: Regular cosmetic issue / Browser-specific minor functionality. Examples: images show in wrong place, or a form looks bad but it still works alright.

**Priority 4**: Copy issues / Minor cosmetic issues / alignment issues

**Priority 5**: Suggestions / Feedback

Once an issue has been submitted you can make comments/updates as needed to the task.

**Note:** Producers can also set priority levels for issues depending on the severity of the issue as they (or their clients) see fit. The priority list above is our QA standard list of priorities where a Producer has not yet intervened on a specific issue(s).

When an issue has been assigned back to QA as ‘Retest’, there are a few different options to select.

* If the issue is fixed you should mark it as closed. If an issue is not fixed you will reopen it and make a comment if there are any updates.
* If an issue is closed on the QA environment, but reappears on the UAT environment you will need to reopen the issue and make a note that it only happens on UAT.
* If the issue is closed on QA, but reappears slightly different on UAT you will need to open a new issue at that point since it’s not the very same issue.
* Depending on the project you will need to use different statuses. Some require issues to be checked on each environment. In this case you will use ‘Passed in Test’/’Passed in UAT’ when an issue is closed.
* ‘On hold’ should be used when an issue can’t be tested because it is blocked from being tested by another issue.
* An issue can be deferred by the PM for any reason, and it is then up to the QA tester to close the deferred issue.
* If for any reason duplicate issues exist, the QA tester will need to close one of them and remove it from the open issues list.

**Testing reference (QA Responsibilities)**

1. Test cases creation.
2. **Smoke testing vs. Sanity testing:**

* When smoke testing one basically asks himself is the site QA test worthy? Once a build gets pushed to QA environment for testing you run quickly through the core components of the site to ensure they are functionally sound—if they are then continue testing; if they aren’t then immediately the site should go back to the developer for further and proper coding. An example: if the Developer completes his coding and the job is then passed on to QA for us to start testing and we begin by testing the Age Gate of a sweepstakes site and the core functionality of the Age gate is broken, wherein it doesn’t work or won’t submit then we immediately send it back to the coder as the site is not QA worthy of testing yet.
* When running a Sanity test we are ensuring that everything appears exactly the same from one environment to the next environment. For example, if testing has been completed on the QA environment and the site build was pushed to UAT we then do a Sanity test to ensure that everything on UAT is matching exactly as that of QA (since the QA environment was already approved and passed testing). If UAT doesn’t match with QA then it fails the Sanity test.

1. **Creative QA:** proofreading, alignment issues, image issues, comparing creative comps against actual creative, etc.
2. **Creative QA cross-browser testing:** same as above but on all supported browsers.
3. **Functional QA:** clicking links and ensuring they target to the correct location, age gate functionality works correctly, Call-to-action buttons target correctly, authentication fields submit correctly, data capture fields work correctly, data capture fields capture correct information in the database, navigation links target correctly, captchas work correctly, overlays and popups work correctly; every link on every page needs verification.
4. **Functional QA cross-browser testing:** same as above but on all supported browsers.
5. **Tracking (analytics):**

* Keeps a record of how many visits to the site. Pixels that are put on tags that fire in the Headers of the site and give information like: what Operating System the user is on, screen resolution the user is using, the browser they’re using. Verify the tags fire correctly and send off the report to the Developer and Account person.
* 2 ways of tracking analytics: upon load of the page and/or on click of any link
* 3 items to check on tracking
  + UA Codes: Google analytics identifier (specific number)
  + Name or title of the tag
  + Correct name of the website page that the tracking came from
* Tracking tools we use most often are Charles and Google Analytics. Use Firefox as the tracking default to work with Charles.

1. **Issues reporting:** every bug/defect found is entered into the defect tracker (either @task, Jira, or ResultSpace 3, wherever the project details are set up). We assign the defects to the coder so they can make fixes. Refer to the Defect Lifecycle and the @task section above on how the defect priorities are arranged. After each full round of testing the QA engineer should send out a quick summary email of that round’s testing results to all involved on the job—QA Lead, Producer, Developer, etc.
2. **Issues retesting:** once the defects have been fixed and reassigned back to QA we perform another retest to ensure the defect has in fact been fixed. If it has been fixed we close out the defect. If it hasn’t been fixed we reassign back to the developer again.
3. **Regression testing:** When QA environment testing is complete (all issues have been fixed by the coder and verified by QA as fixed) and the build is ready to go to UAT (or Preview environment) this is when QA performs a regression site on the QA environment by running through every test case one last time before the site gets pushed to UAT. We also perform regression testing on the Live environment.
4. **Sanity testing on UAT and Production:** refer to step **b.** above.
5. **Post-production QA support:** if requested by the Account team.

**Environments**

1. Local SVN project gets pushed by the developer up to the Development environment
2. The Build team pushes the project files from Dev up to the QA environment
3. File/site path: Dev 🡪 QA 🡪 UAT or Preview 🡪 Production

**QA Default browsers:**

1. IE 8,9 on Win XP & Win7
2. FF Latest version on Win XP & Win7
3. Chrome Latest Version on Win XP & Win7
4. Safari Latest version on MAC

**QA Tablets & Mobile Devices**

1. Nexus 10 tablet
2. Windows Surface RT tablet
3. iPad 2 tablet
4. Blackberry Z10 mobile
5. iPhone 4
6. iPhone 5
7. Samsung Galaxy SII

**Types of Pages**

There are different types of pages that we test and each has its own items that need to be tested. Sweepstakes, Data Capture, Facebook Apps, Landing Pages, Redesigns, Responsive, and Mobile are the different types, and the items to be tested on each individual type are:

**Sweepstakes** – Age Gates will have a specific date to be tested. You should also test the year before, year of, year after and day before, day of, and day after. State restrictions should be tested when applicable. Valid dates need to be checked (2/31/00, 9/31/00) to ensure they don’t work. Business rules should also be verified for the age gate.

**Data captures** – They come in two different types: contact us forms and a registration form. There need to be standards for data captures such as character limits and restrictions on fields. After a form is submitted the information must be verified in a database. Submission rules must be verified such as required fields. Attempt to generate server errors by submitting the form with incorrect information. A security scan must be scheduled to scan any sweepstakes or site with data captures. The proper form must be filled out and emailed to GSO-STAR@sapient.com.

**Facebook Apps** – These apps can also include a sweeps aspect. There will be a like gate, permissions page, shares/wall posts, and redirects (if someone tries accessing the page without liking it first).

**Landing Pages** – This should be a simple page containing links/CTAs and creative testing (alignment, copy, alt tags, font, rollovers, and overlays).

**Site Redesigns** – Similar testing as Landing Pages, but more total pages.

**Responsive sites** – 1 page that changes layout based on a specific viewport (mobile, portrait tablet, landscape tablet, and desktop).

**Mobile** – Checking redirects, accessible vs. optimization, and application vs. site. If it is a mobile app in addition to regular testing you will need to test installing and upgrading the app. Regular sites only need standard QA.