# Totally QA Guidance

## Prerequisites

Many prerequisites need to be fulfilled before a QA can inject quality in the project outcome.

* QA needs to feel comfortable, empowered and feels secure in raising internal escalations
* QA actively participates (active = asking questions, getting answers) in a kick-off meeting and other discussions

All projects are expected to do a kick-off in two or three stages, giving everyone a chance to get an initial overview and allow a later deep-dive:

1. [Mandatory] Initial Project Kick-off with shallow, project-wide description
2. [optional] A Story-groom session at the beginning of an External Release
3. [Mandatory] Weekly Story-grooming session at the beginning of week (or sprint) to cover all the stories that are to be implemented during the week (or sprint)

## Responsibilities, Authorities & Etiquettes

**QA Responsibility @ Kick-off:**

A QA is responsible to see that the prerequisites (mentioned above) are fulfilled. When found incomplete, he is responsible to (a) either get it done (b) escalate to PM/DM/VP.

Once the prerequisites are fulfilled, then the QA is responsible for writing of Acceptance Criteria.

**QA Responsibility During Development**

During development, the QA is responsible for sharing the AC with the developer for his Story and asking him if the AC are complete.

Once development of a Story is complete, the QA is responsible for verifying if the developer has tested his story according to the AC shared by the QA (how well it was done by Dev and what the Dev cannot test).

A QA has an AUTHORITY to set the story status in the WBS to DevDone (or Resolved). A developer cannot decide or update on his own.

A developer may limit his testing to unit tests. A QA is responsible for completing the testing – including, integration and regression testing, System testing, performance testing, smoke testing on production etc.

A QA has a responsibility to add the bugs found to the bug-tracking list. But only those bugs found AFTER the WBS status was set to DevDone/Resolved can be added to the bug-tracking.

A QA is responsible to raise escalation to PM/DM/VP when the etiquette of “1:1 meeting with Dev plus DevDone status” is not followed.

**QA Responsibility for Additional AC**

A QA is responsible to add late-discovered AC to stories and workflows even after the status is set to DevDone.

A QA is responsible for converting (functional) bugs found after DevDone status to additional AC. The attitude has to be “…this bug was found in-spite of following the 1:1 meeting and DevDone status because the original AC was incomplete. So, I need to add a new AC to prevent this bug from being missed in the next round of regression testing…”.

A QA is similarly responsible for converting a customer found (functional) bug to one/many AC that will cover the bug.

**Developers:** A developer usually knows more things about a story (especially technical details) than a QA. A developer’s is responsible to advise QA on how to complete the AC for his stories after reading the AC shared with him.

A developer **MUST** test own work. This testing **MUST** cover all the AC for the story that is programmed. How testing is done **MUST** be shown to the QA.

**PM/BA/Tech PM:** A PM has a responsibility to empower the QA so that (a) knowledge is shared equally with the QA (b) Changes/clarifications are also shared with QA via “documents” (and not via verbal means or grapevine) (c) act on the escalation by QA about non-compliance by developers about the 1:1 etiquette.

When a customer is very particular about the UI, a PM has extra responsibility of organizing a UI/UX review of the implementation before a customer demo or a release. The QA needs to sit in this review and understand what he/she did not catch. The review will be done by a person from Experience Studio.

This document is to be used internally by Extentia members. It should not be shared with external agencies like customers/prospects.

## Acceptance Criteria Tabulation

When all acceptance criteria are written in a single sheet (with a module name column), it helps in getting statical information via pivot tables. Splitting different modules in different tabs makes it difficult to do statistical analysis. However, for a very large project, such a split might be necessary. WHEN SPLIT – it is advisable to split as Admin/Consumer tabs.

The following columns are mandatory in an AC sheet: ID, Module, Page, Story, AC, Type [UI/Functional/Data/?], TestData/Comment

* ID: The ID should be alpha-numeric. Alpha part should be unique for each module. When reporting a bug, a QA can (optionally) mention this ID in the bug report.
* Module, Page/Screen, Story: These are for cross-reference. These names are invented/discovered by a BA when creating a MM, carried forward exactly as in MM to WBS and StoryMatrix (APIMatrix). The same names need to be used even within AC tabulation (uniformity “find in files).
  + Plus “Workflow Name” – column for workflow/scenario tests
* AC: Each story can be covered by multiple AC
  + Each AC needs to have a format similar to “Verify that <some observation> [when <precondition about user-role, specific-data or context>]. The “when” clause is optional.
  + Expects the QAs to use commonsense/knowledge/experience to think of valid steps. The actual steps taken while testing are mentioned only in the bug report and not in AC.
* Actor: An Actor is defined as a role, who interacts with the system within the context of the acceptance criteria. Actor examples are Admin, Consumer, Guest user
* AC Type: UI, Security, Functional, Data – classification needs to be done.
  + UI tests should not try to describe each UI element. This is sufficient: “Verify that the screen is displayed EXACTLY as the approved Visual Design and navigation according to IA”.
  + Following needs to individually verified at element level if (a) there are validation rules -33(b) Display/Hide conditional logic (c) Enable/Disable conditional logic (d) colour change conditional logic
  + Security Tests should include verification of (a) Can a page be loaded by non logged-in user (b) User without proper role (for example: consumer loading admin page) and (c) If data is hidden based on logged-in user role (for example: a consumer can see past purchase orders but the orders of another consumer cannot be displayed by a consumer role. Yet, the business rule may say “admin can see past orders of any consumer”)
  + All the functional AC that do not mention specific data are of type “Functional”. Those AC that specifically mention which data to be used are “Data” tests.
* TestData/Comment: Additional information about AC can be provided

Hundreds of lines of guidance can be written. However, it is unlikely that the QA will even remember guidance that is longer than two pages. That is why the above guidance is kept at a minimum. Story it has crossed two pages.

## Appendix/Examples

AC for Dropdown

* Verify that rows from <source API/table> are populated in the dropdown in <column-name> sequence
* If there is a row-filtering data condition, write separate verification tests

AC for Lists

* Verify that the columns show accurate data (titles & data are not swapped)
* Verify that the lists show correctly filtered rows (does this need “specific data”?)
* Verify that the list is sorted <by what columns>

AC for Details Screen

* Verify that the data fields save and retrieve/show accurate data
* Question: Can such a generic test be understood by QAs? “Verify for Nulls, Data-Type and Data-Length in all the data fields according to the UI”. If the answer is yes, it will save a lot of typing time for the QAs. This is what is meant by EXPECT QAs to use common-sense to interpret such trivial tests.

AC for Long Workflows/Scenarios

* These tests are very important. But QA should not expect developer to test them.
* The AC tabulation format should be similar, except for one additional column after ID “Workflow Name”.
* As far as possible the Module Name or the Page Name + Story Name needs to be specified in each workflow/scenario AC.

## Appendix/Bug Reports/Test History

Bug severity i.e. Blocker, Critical, Major and Minor to be identical in across all the projects

* + Blocker = Drop everything. Fix this now, give me a build
  + Critical = Do not give me next weekly build without fixing this bug
  + Major = This can be prioritized to next week – but must be done in the current release.

Regression suit (subset of AC’s) with the release date and QA status column should contain 3 statuses i.e. Pass, Fail and Blank (the ACs that are not relevant for this release)

A history of test results (file name contains date) may have to be maintained for each sprint. We may have to share this with a customer.

A bug report needs minimum of the following information

* Either Module/Screen/Story correlation or a Test ID that failed.
* Steps and data to reproduce consistently
* Expected and Actual Result

This kind of “structured thinking” may help in reporting a bug that a programmer understands… GIVEN <precondition> WHEN <user does something> THEN <I observe actual-result, whereas I expect expected-result>

## Appendix/Release Notes

It makes sense to share the Test suit and the list of known bugs with the release notes.

* One phrase that annoys customers is “We already know this bug exists”. Mention known bugs in the Release notes and then never ever say that sentence.

## Appendix/Postman Tests

* Minimum performance and security tests to be performed and shared with higher management.
* API’s need to be automated in Postman and following minimum assertions to be covered
  + Global variables for environment URL root need to be maintained.
  + Size of response (below ???) and response time (below 1s?) should also be tested.
  + The 200, 201 400, 401, 403, 404 and 500 status code testing should also be included in Postman tests.
  + Can data type, length, not null validation be included in the postman automation?
  + Persisted data verification (what is read is what was stored) needs to be part of the Postman collection [a.k.a server returns a predictable value based on the input condition]
  + Automating of 3rd party API tests are mandatory [do 3rd party API work according to what they claim?]

## Appendix/Kick-off Meeting

It is also implicit that Sprint 0 needs to be completed with good quality before the kick-off meeting is held.

* The following are discussed:
  + The project’s Functional Requirements with the help of a MindMap, Visual Designs, IA, SOW/BRD, Work Breakdown in Estimation Sheet
  + The technical solution approach with the help of API-Matrix (StoryMatrix), ER Diagram, Topology (including 3rd party system interactions)
  + Individual Stories
  + Technical risks
  + Dependencies (on customer, on other Extentia teams, on 3rd parties)

## Additional points

Postman videos

Postman: YouTube videos  
 <https://www.youtube.com/watch?v=1FxKWHeAcDs&list=PLIMhDiITmNrJgUekucfx3X7XxyekZHxJj&index=5>   
<https://www.youtube.com/watch?v=qDb7v9MrQ38> <https://www.youtube.com/watch?v=kLCviw_9Hcc>   
Advanced Stuff: <https://www.youtube.com/watch?v=Ayo_KdLLcTA&list=PLM-7VG-sgbtBE0mqMBlAYNuqTkhJzFjlP&index=1>

How the test the release with cherry-pick - **TBD**

Below are the queries that I have:-

1. An example of a blocker that generally occurs in every project.
2. Is it fine to continue the testing even if the critical bug is affecting other functionalities and in result other test cases are also getting affected?
3. Can major bug be a critical bug? Can you please explain what is major and what is critical?
4. Weakly build that is given by the developer side, and release comes under deliverables?
5. Weekly grooming sessions and kick -off meeting are same?
6. Can I go through how to make SOW and IA.
7. Internal dependency means it is dependent on the team and external dependency is on audience outside of the team?
8. If kick off meeting does not happen, then how QA can make it happen?
9. Acceptance criteria is given by BA then why QA should add more and then Developer?
10. What is the limit of number of AC or the set of rules to write AC?