# QA

## General Checklists

The checklists includes the check points which contains when QA needs to consider when proceeding API, Web, Mobile Testing.

### Test Case writing - General guideline

|  |  |
| --- | --- |
| **Focus Criteria** | **Expected coverage** |
| **Understand Requirements** | Thoroughly understand the requirements and functionality of the software. |
|  | Clear understanding of what needs to be tested and the expected outcomes. |
|  | Get the Clear Idea of acceptance criteria and points described |
|  | Analyze and Clarify the requirement in grooming sessions |
|  |  |
| **Test Scenario Planning** | Plan the Test scenarios by reading acceptance criteria |
|  | Do not copy acceptance criteria as it is for the test case |
|  | Focus on positive and Negative functionalities |
|  | Include test cases for boundary and edge conditions. |
|  | Test the limits of the system's capabilities to identify potential issues. |
|  | Include negative test cases to validate error handling and system responses to invalid inputs. |
|  | Check how the system behaves under unexpected or erroneous conditions. |
|  |  |
| **Use a Standard Template** | Use AIO for writing test cases |
|  | Follow the AIO format |
|  |  |
| **Test Case Title** | Use descriptive, short and unique titles for test cases.(Do not write lengthy titles) |
|  | Titles should indicate what functionality or scenario is being tested. |
|  | Title should start with "Verify the" or "Verify whether" (Rarely we can drop these as per the test scenario covering) |
|  |  |

|  |  |
| --- | --- |
| **Test Case structuring** | if it is a UI related test then start the test with the tag [UI] then module name [Module Name] then verification |
|  | **Sample** - [UI][Search Field]- Verify whether the Search option displays next to search Bar field |
|  | If it is functional test case then [Module Name] [Sub Module Name]- Verification (functional tag is not adding to functional test cases since most of the test cases are to cover functional) |
|  | **Sample** - [Search Field]- Verify the search result by entering a search term with special characters |
|  | If it is Non functional then [NonFunctional] [Module Name] then verification |
|  | **Sample** - [Non Functional][Login]- Verify the application behavior when login user count >100 |
|  | Should follow the AIO Folder structure |
|  |  |
| **Test Preconditions** | Clearly state any prerequisites or preconditions necessary for executing the test case. |
|  | Ensure the system is in a specific state before the test case is executed. |
|  | If it is highlighted Pre Req the better to add it as the first step |
|  | **Sample - Pre Req-** User should be logged in as a admin to view the "Specific feature name" |
|  | If it is general pre requisite then better to add it in pre req field in AIO |
|  | Do not cover main steps in pre requisites |
|  |  |
| **Test Steps** | Write detailed step-by-step instructions to reproduce the test scenario. |
|  | Include specific input data, actions to be performed, and the expected outcomes. |
|  | Do not write more and more steps to cover the navigation steps-either cover it by pre req or maximum 1 or 2 steps |
|  | Always write the test steps by thinking the executor has no knowledge in Domain |
|  |  |

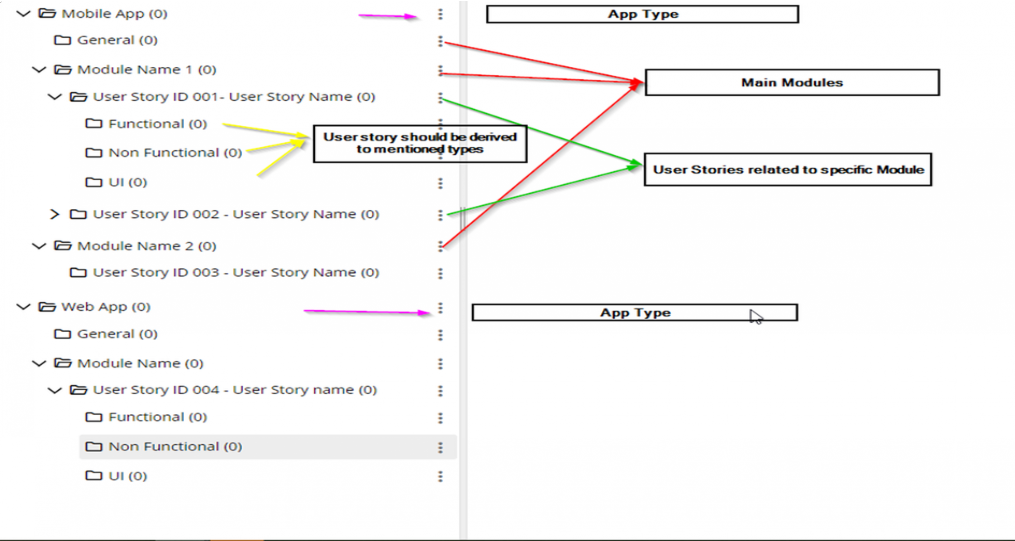
|  |  |
| --- | --- |
| **Test Data** | Always try to add test data or SQL query to preparation test data |
|  | Always try to validate data with DB |
|  | Handle data dependencies and ensure data consistency across test cases. |
|  |  |
| **Expected Result** | Clearly define what the expected results should be after executing each step. |
|  | Be specific about the outcomes, including any messages, errors, or visual cues. |
|  |  |
| **End Verification** | Always last step should be a verification step and it should matches the title |
|  |  |
| **Reusability** | Write test cases that are modular and can be reused for regression testing. |
|  | Avoid hard-coding values whenever possible. |
|  |  |
| **Review and Revision** | Have test cases reviewed by peers or stakeholders to ensure accuracy and completeness. |
|  | Revise and update test cases as the application evolves and requirements change. |
|  | Focus on previous given comments/feedback when writing test cases |
|  | Do not keep test case just moving to under review. Contact or communicate to someone responsible or share a report with leads |
|  |  |
| **Communication** | Communicate effectively with the product owner/ Dev leads to clarify doubts and resolve issues. |
|  | Foster a collaborative environment between QA and development teams |

##### AIO Folder structure

AIO Confluence page Ref-



[AIO Tests - All-In-One Test Management for Jira](https://aiosupport.atlassian.net/wiki/spaces/NAT/overview)



|  |  |  |
| --- | --- | --- |
| **Folder type** | **Description** |  |
| Module Name | Main Module where user stories related to |  |
| User story Id | Jira User Story ID |  |
| User Story Name | User story title or covering feature name |  |
| Functional | All the Functional Test Cases Shoud be added in this folder |  |
| Non Functional | All the Functional Test Cases Shoud be added in this folder |  |
| UI | All the UI Test Cases Shoud be added in this folder |  |
|  |  |  |
| Note | If there are no specific module then create a folder call General and add test cases in to that | If the application has both mobile app and web app and API coverage then create different folders to maintain test cases |

### API Testing Checklist

|  |  |  |
| --- | --- | --- |
| **Focus Area** | **Test scenario** |  |
| **Functional Testing** | **Endpoint Testing** | Verify that each API endpoint functions correctly and returns the expected responses. |
|  | **HTTP Methods** | Test all supported HTTP methods (GET, POST, PUT, DELETE, etc.)  for each endpoint. |
|  | **Request Parameters** | Test various combinations of request parameters, including required, optional, and edge cases. |
|  | **Authentication and Authorization:** | Ensure that authentication and authorization mechanisms work as intended. |
|  | **Status Codes** | Verify that the API returns appropriate HTTP status codes for different scenarios (e.g., 200 OK, 400 Bad Request, 401  Unauthorized, 403 Forbidden, 404  Not Found, 500 internal server error). |
|  |  |  |
| **Data Validation** | **Input Validation** | Check how the API handles invalid input data, such as missing or incorrect parameters. |
|  | **Data Integrity** | Verify that data sent and received remains intact and unaltered during transmission. |
|  | **Data Formats** | Ensure that data is in the expected format (e.g., JSON, XML, etc.) and adheres to defined schemas. |
|  |  |  |
| **Error Handling** | **Error Responses** | est error responses to ensure they provide useful and meaningful error messages. |
|  | **Exception Handling** | Check how the API handles unexpected errors and exceptions. |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Security Testing** | **Injection Attacks** | Test for common security vulnerabilities like SQL injection and Cross-Site Scripting (XSS). |
|  | **Authentication and Authorization** | Verify that only authorized users can access certain endpoints and data. |
|  | **Rate Limiting** | Test if rate limiting and throttling mechanisms are in place to prevent abuse. |
|  |  |  |
| **Integration Testing** |  | Test how the API interacts with other components or third-party services it relies on. |
|  |  | Ensure proper data flow between integrated systems. |
|  |  |  |
| **Regression Testing** |  | Continuously run tests to detect any regressions or issues introduced by code changes. |
| **Version Compatibility** |  | Ensure backward and forward compatibility with different versions of the API |
| **Documentation Testing** |  | Ensure that the API documentation is accurate, up-to- date, and aligns with the actual API behavior. |
| **Environmental Factors** |  | Test the API in different environments (e.g., development, staging, production) to catch environment-specific issues. |

### UI/UX Testing Checklist

UI checklist has been divided into the categories UI Testing, Ux Testing, Web responsive test and mobile responsive test

#### UI Checklist

|  |  |
| --- | --- |
| **Area** | **Conditions** |
| **Validation messages** | Mandatory fields validation |
|  | Characters validation |
|  | Special characters validation |
|  | Number validations |
|  | Specific messages for difference scenarios |
|  | Validate that error messages are displayed clearly and help users understand how to resolve issues. |
|  |  |
| **Spellings** | Check for correct capitalization |
|  | The spellings are accurate and according to relevant language in each UI Component |
|  |  |
| **Consistency** | Color codes are correct |
|  | Font style is according to the requirement |
|  | Font size is according to the requirement |
|  | Labels are according to the requirement |
|  | Line spacing should be consistent |
|  | Alignments are consistent |
|  | Positioning is correct |
|  | Date format is according to the requirement |
|  |  |
| **Images** | Image quality is sufficient |
|  | Support for the different formats |
|  | Cropping option are available or not |
|  |  |
| **Browsers** | Check if the UIs are according to the Figma in different browsers |
|  |  |
| **Overlapping** | Overlapping should not happen among UI component |
|  |  |

|  |  |
| --- | --- |
| **Icons and Logos** | Check for the icons and logos placement |
|  |  |
| **Header** | Header should be according to the requirement |
|  |  |
| **Footer** | Footer should be common for all the pages |
|  |  |
| **Disabled/Enabled** | Check for the disabled/enabled behavior of every UI component |
|  |  |
|  |  |
| **Tooltip** | Verify tool tip is available when hovering to the UI component |
|  |  |
| **Resolution** | Verify UI behavior under different resolution |
|  |  |
| **Zoom** | Entire UI should not change after zoom in or zoom out |
|  | Verify the Screen appearance in Full screen mode |
|  |  |
| **Navigations** | Forward navigations should be able to to based on permissions |
|  | Backward navigations should be able to do based on permissions |
|  |  |
| **Scrollbars** | Both horizontal and vertical scroll bars should appear based on page sizes |
|  |  |
| **Mouse & Keyboard** | Should be able to operate using Keyboards |
|  | Should be able to operate using mouse |

#### UX Checklist

|  |  |
| --- | --- |
| **Area** | **Condition** |
| **Usability** | Evaluate the overall ease of use and user- friendliness of the UI. |
|  | Test common user tasks to identify any bottlenecks or confusing elements. |
|  |  |
| **Loading and Performance** | Check loading times for various UI components. |
|  |  |
|  | Test performance under different levels of user load. |
|  |  |
| **Feedback and Progress** | Ensure that users receive appropriate feedback when actions are performed (e.g., success messages, loading spinners). |
|  | Test long-running processes and confirm that users are informed of progress. |
|  |  |
| **Localization** | Test the UI with different language and regional settings to ensure text translations and formatting are accurate. |
|  |  |
| **Accessibility** | Check if the UI follows accessibility guidelines (WCAG) for users with disabilities. |
|  | Test with screen readers and keyboard navigation. |
|  |  |
| **Consistency** | Verify that design elements, language, and behaviors are consistent throughout the application. |

#### Web Responsive checklist

Responsive design, in the context of web apps, refers to the approach of designing and developing applications in a way that allows them to adapt and provide an optimal user experience across various devices and screen sizes. This includes desktop computers, laptops, tablets, and smartphones. The primary goal of responsive design is to ensure that the web app's layout, content, and functionality remain usable and visually appealing regardless of the device being used to access it.

|  |  |
| --- | --- |
| **Context** | **Description** |
| **Fluid Grid Layouts** | This ensures that elements on the web app's interface can adjust their size and position based on the screen width.uses percentage-based widths instead of fixed widths, |
| **Flexible Images and Media** | Images and media elements are scaled proportionally to fit different screen sizes. This prevents images from being too large or too small on different devices. |
| **Media Queries:** | Media queries are CSS rules that allow you to apply styles based on various characteristics of the device, such as screen width, height, and orientation. These queries enable you to customize the appearance of your web app for different devices. |
| **Viewport Meta Tag** | The viewport meta tag allows you to control how your web app is displayed on different devices. It lets you set the initial scale, maximum scale, and other viewport-related properties. |
| **Content Prioritization** | In responsive design, you often need to prioritize certain content on smaller screens due to limited space. This might involve reorganizing elements, hiding less important content, or collapsing menus into a mobile-friendly format. |
| **Touch-Friendly Interfaces** | Since mobile devices rely heavily on touch interactions, responsive design often includes optimizing UI elements for touch gestures, like swiping and tapping. |
| **Testing Across Devices** | Responsive design involves thorough testing on various devices, screen sizes, and orientations to ensure a consistent and pleasant experience for all users. |
| **Progressive Enhancement** | This design philosophy involves starting with a basic version of the web app that works on all devices, and then progressively adding more advanced features and styling for larger screens. |

Ref- .



[Responsive Web Design: What It Is And How To Use It — Smashing Magazine](https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#%3A~%3Atext%3DResponsive%20Web%20design%20is%20the%2Cuse%20of%20CSS%20media%20queries)

#### Mobile Responsive checklist

Responsive design in the context of mobile apps refers to the approach of designing and developing mobile applications in a way that allows them to adapt and function well across a variety of devices and screen sizes. The goal of responsive design is to provide a seamless user experience regardless of whether the app is being accessed on a smartphone, tablet, or other mobile devices.

|  |  |
| --- | --- |
| **Context** | **Description** |
| **Flexible Layouts:** | Designing layouts that can adjust to different screen sizes and orientations. This involves using fluid grids and relative units for elements like fonts and spacing, so they scale proportionally. |
| **Adaptive User Interface (UI):** | Modifying the user interface based on the available screen space. This might involve rearranging or reorganizing elements, changing navigation patterns, and adapting the content hierarchy. |
| **Media Queries:** | Utilizing CSS media queries to apply different styles and layouts based on the device's characteristics, such as screen width, height, and resolution. This allows developers to create specific designs for various breakpoints. |
| **Touch-Friendly Interactions:** | Designing interactive elements and controls that are easy to tap, swipe, and interact with using touch gestures. This includes ensuring that buttons are appropriately sized and spaced to accommodate fingers of different sizes. |
| **Optimized Images and Assets:** | Delivering images and media at appropriate resolutions and formats based on the device's capabilities. This helps to balance visual quality with loading times. |
| **Progressive Enhancement:** | Building the app's functionality in layers, so that even on devices with limited capabilities, the core functionality is still accessible and usable. |
| **Performance Optimization:** | Paying attention to performance considerations, such as minimizing unnecessary animations, reducing server requests, and optimizing code for faster loading times. |
| **Cross-Device Testing:** | Thoroughly testing the app on a variety of devices, screen sizes, and orientations to ensure consistent performance and usability across the board. |

### Mobile Testing Checklist

This has been divided into three main categories to make things more clear. The main categories are UI, Functional and General

#### UI - Checklist

|  |  |
| --- | --- |
| **Verifications** | **Category** |
| Check whether all the mandatory fields are indicated as per requirement | **Validation** |
| Check whether app validate mandatory field when left blank |
| Check whether validation message shown (If applicable) |
| Check whether app validate input data types (If applicable) |
|  |  |
| If message is given : must be compatible with given text | **Must check all the messages (Success, error, warning etc.) text** |
| If message is not specified: Suitable generic message list must be prepared and verify accordingly |
|  |  |
| Check spellings of the text | **Spellings** |
| If UK Project-Spelling must be in UK English |  |
| If US Project - Spelling must be in US English |
|  |  |
| Check consistency of the UI components | **Consistency** |
| 01. Font and font sizes |
| 02. Button label, colour, sizes |
| 03. Field label, colour, sizes |
| Date format (country wise if not specified) |
| 01. Date must be in the specified format |
| 02. IF Date format not specified, date should be formatted according to the project country |
|  |  |
| Check ability to use and access mobile feature (camera, gallery etc) when required (When devise permission granted) | **Mobile Permission** |
| Check restrict access mobile features (camera, gallery etc) (When devise permission not granted) |

|  |  |
| --- | --- |
|  |  |
| Upload different types of images | **Images** |
| check Image quality (Pixelate, resolution) |
| Check whether are there any auto re size, cropper or fit to screen functionality available |
| If cropper functionality available, must be able to adjust |
| Must check with different size images |
| Must check image quality after crop |
|  |  |
| Must check text compatibility with design | **Text** |
| Font size |
| Font colour |
| Font family |
| Text decoration |
|  |  |
| Browser compatibility | **Browse** |
| Inner browser UI compatibility and responsive |
| Email functionalities |
|  |
|  |

#### Functional Checklist

|  |  |
| --- | --- |
| **Verification** | **Category** |
| Check whether app demand unnecessary permission at the begging of installation | **Device permission** |
| (Location permission, subscription etc) |  |
|  |  |
| If product has requirement that user must be able to use app offline(work app without internet ) | **offline behaviors** |
| Must identify what are the features must be able to use offline |
| Must test app functionality and behaviours without internet |
|  |  |
| Check data loading time | **Data Loading** |
| Check average time take to perform a search (Data loading speed/time) |
| Check impact on memory |
| Check maximum data load (up to crash) |
| Loading time |
| crash/time out point |
| Check lazy loading behaviours |
| Check whether data loading happens simultaneously when scroll up/down |
| Check whether relevant data loading happens as expected (text, images) |
| Check whether data loading hinder user experience (smoothness) |
| Note: Use script developed by devs |
|  |
|  |
|  |
|  |  |
| Speed | **Check Uploading Time** |
| app crash point |

|  |  |
| --- | --- |
|  |  |
| Must clarify are there any exception before start testing | **Exception** |
| If there are exception, must test all those scenarios |
|  |
|  |
| Check application loading times | **Performance** |
| This should check device wise | **Gesture functionality must check** |
| Must check smoothness when touch |
|  |  |
| Check whether notification shown for both app types (iOS and Android) | **Notification behaviours must check** |
| Check whether notification shown for both app types at the same time (iOS and Android) |
| Check whether notification shown upon WIFI/mobile data on for both app types (iOS and Android) - when offline |
| Check whether notification shown when app open |
| Check whether notification shown when app close |
| Check whether notification shown when app run in background |
| Check whether Opening app through notification |
| Check whether redirect app feature through notification (if applicable) |
|  |  |
| Block screen (when update release) behaviors must check |  |
|  |  |
| Check whether app allow to send email in invisible mode | **iOS Specific** |
|  |

#### General Checklist

|  |  |
| --- | --- |
| **Verification** | **Category** |
| Test the installation and uninstallation process of the app | **Installation and Updates** |
| Verify that updates or patches do not break existing functionality. |  |
|  |  |
| Should get the app from test environment for testing | **Test Environment** |
| Tester must get the app from below locations for testing |
| iOS - Test flight |
| Android - google Play Console |
| Note: Should not test app from dev build. Tester must get the testing app from the respective test site |
|  |  |
| Check that the app functions on older devices and operating system versions. | **Back Ward Compatibility** |
|  |  |
| Must clarify whether app should allow either dark or light mode or both | **Dark /Light Mode** |
| IF So, |
| Check whether user is able to change the mode |
| Check text colour changes |
| Check text design changes |
| Check component colour changes |
| Must clarify are there any mode must be blocked |
| IF So, |
| Check whether app allow to change mode |
|  |  |
| Check whether are there any requirement to block either Portrait or Landscape mode | **Portrait and Landscape** |
| If so, |
| must identify applicable devices types (smart phone, tab, etc) |

|  |  |
| --- | --- |
| must check this has been implemented |  |
| Check UI in Portrait mode and landscape mode |
| IF Landscaped mode blocked |
| Check whether the device allow landscape mode (app must not auto re size the screen in to landscape( |
| App should not depend on device configuration and must not allow landscape |
|  |  |
| Email password or OTP Login | **Check whether app allow to sign in with following methods (if appliable)** |
| Social login (google, fb, apple etc) |
| finger print |
| facial recognition |
|  |  |
| Check social logo, icon and text when use social sign in / sign up | **Check apple logo and text when use apple sign in / sign up** |
| Follow all the relevant guideline specified by the org. |
|  |
|  |
|  |
|  |
| Android | **Must test on multiple devices** |
| Smartphone |
| Tablet |
| iOS |
| iPhone |
| iPad |
| Standard screen sizes for testing |
| Android |
| Version : API Level 21 / Android 5 (Lollipop) or later version |
| iOS |
| Version : iPhone 11 or later version |
| *Note: mention the tested and agreed devices and* |

|  |  |
| --- | --- |
| *tested scenarios* |  |
| 01. We agreed app will support iPhone 11 or later version |
| 02. We use all the devices what we have and others check simulators and emulators in oder to cover maximum device types and sizes |
|  |  |
| Must check following screen sizes (according to device availability) | **Screen Sizes** |
| ex- iOS |
| 13 |
| 13 Mini |
| 13 Pro |
| 13 Pro max |
| ex- Android devices |
|  |  |
| Privacy policy | **Check Policies and other statements (if applicable)** |
| Terms and condition |
| Other |
|  |  |
| Version of the application (Release management) | **App Version** |
| Should check version number of the app (Version number must be mentioned in the app) |
| 01. Check the place where version mentioned in the app |
| 02. Check the app store |
| Check whether the version number is defined according to the release (Build, enhancement or bug fix) |
|  |  |
| Tested scope and tested device types and sizes must be mentioned in the release note | **Release note** |
| Release note must include following information |
| Devices used for testing (Resolution and screen size) |
| Test scope |

## Automation

### Web Automation

### API Automation

### Mobile Automation

### NFRs

Automated NFRs testing planning and technologies to be defined and setup.

## Defect handling

Pre-done bugs.

 create failing test cases. UAT bugs.

 create a bug ticket.

 tag the component field. Production bugs.

 create a bug ticket.

 tag the component field.

Pair testing

 e2e bugs.

 create a story.

 set ceiling value to limit the scope.  add bug tickets as test cases.

 #define a workflow  Pre-done stories.

Once UAT and Production bug received

* 1. Proper investigation should be done and needs to share a postmortem report to the team

## QA Deliverables

Within the Sprint

 Test Cases written for specific user stories (AIO should up to date)

 Test Execution summary for each sprint (AIO should up to date with Cycles)  Sprint Review session (end of the sprint)

 Sprint Summary report/ Sprint Release note

 Automation Progress report (If project implemented the automation)

Major release to a client

This release note should be shared when a major release / After CR implemented or After client bug fixes

QA Release Note

After UAT/Production Feedback

 Postmortem for the UAT Bugs

 Postmortem for the production bugs

### Deliverable Formats

The sample format should be customized as per the project

**Sample Sprint Summary Report/ Internal Release note**



**Sprint - Relase … te.docx**

09 Nov 2023, 02:56 AM

**Sample Client Release Note**



**Release Note.docx**

08 Nov 2023, 07:55 AM

**Postmortem update for UAT/Production bugs**



**Postmortem Up…te.docx**

08 Nov 2023, 08:03 AM

# QA - Tools

AIO

Test Automation Framework Load testing tools

Security testing tools.