**1. Test Case: Verify the Page Title**

**Objective**: Ensure the title of the page matches the expected title.

* **Test Case ID**: TC\_UI\_001
* **Test Case Name**: Verify the page title
* **Test Steps**:
  1. Open the homepage of **awesomeqa.com/ui**.
  2. Check the page title in the browser tab.
* **Expected Result**: The page title should be “AwesomeQA UI Testing”.
* **Pass/Fail Criteria**: The page title should exactly match “ABC”.

**2. Test Case: Verify Header Elements**

**Objective**: Ensure the header contains the necessary elements (Logo, Navigation Bar).

* **Test Case ID**: TC\_UI\_002
* **Test Case Name**: Verify header elements
* **Test Steps**:
  1. Open the homepage of **awesomeqa.com/ui**.
  2. Check for the presence of the logo in the header section.
  3. Verify that the navigation bar has links such as Home, About, Contact, etc.
* **Expected Result**:
  1. The logo should be present in the header section.
  2. The navigation bar should contain Home, About, and Contact links.
* **Pass/Fail Criteria**:
  1. If the logo and navigation links are missing or broken, the test fails.

**3. Test Case: Verify Login Functionality**

**Objective**: Ensure the login form accepts valid credentials and rejects invalid ones.

* **Test Case ID**: TC\_UI\_004
* **Test Case Name**: Verify login functionality
* **Test Steps**:
  1. Navigate to the login page.
  2. Enter a valid username and password.
  3. Click the **Login** button.
* **Expected Result**:
  1. After clicking **Login**, the user should be redirected to the dashboard or homepage.
* **Pass/Fail Criteria**:
  1. If the user is not redirected or sees an error message, the test fails.

**4. Test Case: Verify Search Bar Functionality**

**Objective**: Ensure that the search bar on the homepage works as expected.

* **Test Case ID**: TC\_UI\_005
* **Test Case Name**: Verify search bar functionality
* **Test Steps**:
  1. Navigate to the homepage.
  2. Type a search term (e.g., "UI Testing") into the search bar.
  3. Press Enter or click the search icon.
* **Expected Result**:
  1. The search results page should display relevant results based on the search term entered.
* **Pass/Fail Criteria**:
  1. If no results are shown or incorrect results are displayed, the test fails.

**9. Test Case: Verify Image Display**

**Objective**: Ensure that all images on the page load correctly and are not broken.

* **Test Case ID**: TC\_UI\_009
* **Test Case Name**: Verify image display
* **Test Steps**:
  1. Open the homepage and check for the presence of images (e.g., banners, product images).
  2. Check if each image loads properly without any broken image icons.
* **Expected Result**:
  1. All images should load without any issues.
* **Pass/Fail Criteria**:
  1. If any image does not load or shows a broken link, the test fails.

**10. Test Case: Verify Page Load Time**

**Objective**: Ensure the webpage loads within an acceptable time frame.

* **Test Case ID**: TC\_UI\_010
* **Test Case Name**: Verify page load time
* **Test Steps**:
  1. Open the homepage.
  2. Measure the time it takes for the page to fully load.
* **Expected Result**:
  1. The page should load within 3-5 seconds.
* **Pass/Fail Criteria**:
  1. If the page takes longer than 5 seconds to load, the test fails.

**Conclusion:**

This exercise provides a set of basic UI test cases that cover common elements and interactions found in most websites. You can adapt these to the specifics of the **AwesomeQA.com/UI** website, or any other site you’re testing, to ensure proper functionality and a smooth user experience.

4o mini

**You said:**

Please provide the basic level practice exercise based on awesomeqa.com/ui site for writing test cases , executing test cases , creating test cycle , executing test cycle , defect tracking , traceability matrices using Jira , Zephyr squad for beginners level . I want to share these exercises to my training associate to practice jira , zephyr. At least I need 5 to 6 different exercises .

**ChatGPT said:**

ChatGPT

Sure! Below are **5 basic-level practice exercises** designed to help your training associates learn how to use **Jira** and **Zephyr Squad** for test management. These exercises will help your team practice writing test cases, executing test cases, creating and executing test cycles, defect tracking, and traceability matrices in **Jira** and **Zephyr Squad**. The exercises are geared toward beginners and can be performed using the sample site, **awesomeqa.com/ui**.

**Exercise 1: Writing Test Cases in Zephyr Squad (Jira)**

**Objective**: Learn how to write test cases in Zephyr Squad.

**Steps**:

1. **Access Zephyr Squad**:
   * Login to Jira and open the **Zephyr Squad** plugin.
   * Navigate to the "Tests" section.
2. **Create Test Cases**:
   * Write 3 test cases based on the **awesomeqa.com/ui** site. The test cases should cover different aspects of UI testing such as:
     + **Verify the page title**.
     + **Verify contact form validation**.
     + **Verify the login functionality**.

**Expected Deliverables**:

* Each test case should contain:
  + Test Case Name
  + Test Steps
  + Expected Results
  + Pass/Fail Criteria
  + Test Priority (e.g., High, Medium, Low)

**Example Test Case (for "Verify the page title")**:

* **Test Case ID**: TC\_UI\_001
* **Test Case Name**: Verify Page Title
* **Test Steps**:
  1. Open **awesomeqa.com/ui** homepage.
  2. Check the page title in the browser tab.
* **Expected Result**: The page title should be "AwesomeQA UI Testing".
* **Pass/Fail Criteria**: The test passes if the page title matches; otherwise, it fails.

**Exercise 2: Executing Test Cases in Zephyr Squad**

**Objective**: Learn how to execute test cases and mark the status in Zephyr Squad.

**Steps**:

1. **Create a Test Execution**:
   * Go to the **Zephyr Squad** plugin in Jira.
   * Create a **Test Execution** for the test cycle by selecting the relevant test cases (e.g., from Exercise 1).
   * Link them to a **Jira version** (for example, **v1.0**) or a specific sprint if relevant.
2. **Execute the Test Cases**:
   * Mark each test case as **Pass** or **Fail** based on the actual results observed on the **awesomeqa.com/ui** site.
   * Capture the **Execution Status** for each test case (Pass/Fail/Blocked/Skipped).

**Expected Deliverables**:

* The test execution should show the status of all test cases (Pass/Fail/Blocked).
* Provide comments or attachments (if needed) to clarify test results.

**Example Execution**:

* **Test Case**: Verify the page title (TC\_UI\_001)
* **Execution Status**: Pass (if the title matches "AwesomeQA UI Testing")
* **Comments**: "Page title correctly displayed as expected."

**Exercise 3: Creating Test Cycles in Zephyr Squad**

**Objective**: Learn how to create and manage a **Test Cycle**.

**Steps**:

1. **Create a Test Cycle**:
   * In Zephyr Squad, go to the **Test Cycle** section.
   * Create a new test cycle, name it as **"UI Testing Cycle v1.0"**, and associate it with the version **v1.0** or a specific sprint.
   * Add the test cases from **Exercise 1** to the test cycle.
2. **Schedule the Test Cycle**:
   * Assign the test cycle to specific team members (your associates can act as testers).
   * Set start and end dates for the test cycle.

**Expected Deliverables**:

* A test cycle created with all the test cases linked to it.
* Testers (associates) can execute the test cases in this test cycle.

**Example Test Cycle**:

* **Test Cycle Name**: UI Testing Cycle v1.0
* **Version**: v1.0
* **Test Cases Included**: TC\_UI\_001, TC\_UI\_002, TC\_UI\_003, etc.

**Exercise 4: Executing a Test Cycle**

**Objective**: Learn how to execute a complete **Test Cycle** in Zephyr Squad.

**Steps**:

1. **Go to the Test Cycle**:
   * Navigate to the **Test Cycle** you created in Exercise 3 (e.g., **UI Testing Cycle v1.0**).
   * Open the test cycle and review the test cases added to it.
2. **Execute Test Cases**:
   * As a tester, execute each test case within the cycle.
   * For each test case, update the status to **Pass**, **Fail**, **Blocked**, or **Skipped** based on the results.
   * Add any comments if necessary, such as if the test case was blocked due to an environment issue or dependency.
3. **Complete the Cycle**:
   * Once all the test cases have been executed, mark the test cycle as **Complete**.

**Expected Deliverables**:

* A completed test cycle with execution statuses updated.
* Comments or attachments explaining issues or blockers.

**Example**:

* **Test Cycle**: UI Testing Cycle v1.0
* **Test Execution**: Each test case executed with status (Pass/Fail).

**Exercise 5: Defect Tracking in Jira**

**Objective**: Learn how to log and track defects related to test failures in Jira.

**Steps**:

1. **Log a Defect**:
   * After executing test cases in the test cycle (Exercise 4), if any test case fails, log a **defect** in Jira.
   * Create a new **Jira Issue** with the defect type (Bug) and link it to the test case and test cycle.
   * Provide detailed steps to reproduce, severity, priority, and any screenshots or attachments to help the developer.
2. **Link Defect to Test Case**:
   * Link the defect to the **failing test case**.
   * Ensure the test case status is updated to **Failed** and **Blocked** if the defect prevents the test from passing.

**Expected Deliverables**:

* Jira defect tickets created for any failing test cases.
* Linkage between test cases and defects.

**Example Defect**:

* **Defect ID**: BUG-001
* **Summary**: Login form fails when submitting empty fields.
* **Linked Test Case**: TC\_UI\_004
* **Priority**: High
* **Steps to Reproduce**:
  + Navigate to the login page.
  + Leave all fields empty and click **Login**.
* **Status**: Open

**Exercise 6: Creating Traceability Matrix in Jira**

**Objective**: Learn how to create a **Traceability Matrix** in Jira to ensure test coverage and defect management.

**Steps**:

1. **Create Traceability Matrix**:
   * Create a **Traceability Matrix** using the test cases (from Exercise 1) and the defects (from Exercise 5).
   * The matrix should link each test case to its related requirement or user story (if applicable).
   * It should also link to the defects found in the related test cases.
2. **Populate the Matrix**:
   * List all test cases, their statuses, and any associated defects.
   * Ensure that each defect is linked to the corresponding test case and show whether the test case has been re-executed after defect resolution.
3. **Generate Reports**:
   * Use Jira and Zephyr Squad to generate a traceability report showing the relationship between test cases, defects, and requirements.

**Expected Deliverables**:

* A traceability matrix document or Jira report linking test cases, defects, and user stories/requirements.

**Example Matrix**:

| **Test Case ID** | **Test Case Name** | **Status** | **Defect ID** | **Defect Status** | **Requirement** |
| --- | --- | --- | --- | --- | --- |
| TC\_UI\_001 | Verify Page Title | Pass | None | N/A | REQ-001 |
| TC\_UI\_004 | Verify Login Functionality | Fail | BUG-001 | Open | REQ-002 |
| TC\_UI\_003 | Verify Contact Form Validatio | Pass | None | N/A | REQ-003 |