QA Hiring - Technical Exercises

Introduction

QA Engineers at Atlassian are our (not so) secret weapon. They not only help teams continually improve the quality of the software they create, but they also help teams do it more effectively. In short, our QA engineers are essential for shipping quality software faster.

Our developers know this. QA is routinely voted the most satisfying aspect of how we develop software at Atlassian. As such, developers constantly pull QA engineers into collaborating on their work. This is not a company where code is thrown over the wall to QA, nor a place for people who don't want to be deeply involved in the development process.

To be effective, our QA engineers need strong testing, technical and collaboration skills. The aim of this set of exercises is to allow you to demonstrate some of those.

To complete, please:

1. Create a **private** repository on Bitbucket, Atlassian's code hosting service: https://bitbucket.org/
2. Upload your files to that repository.
3. Under Settings->Access Management for the repository, grant Admin access to the "atlassian-qa" user.
4. Let us know the link to your repository via email.

Exercise 1 - Test Automation

Goal

- To demonstrate your skill at coding automation frameworks for the use of other developers.
- To demonstrate your ability to advise others on writing valuable automated tests.

Task

A team of developers is inexperienced with writing automated tests and have asked for your help to get them started. The target of the testing is Confluence Cloud, Atlassian's hosted wiki product.

To help them, write some example automated tests to verify that a user can:

- Create a new page.
- Set restrictions on an existing page.

The developers will use your tests as a reference to create their own, so be sure that your tests are robust, maintainable and valuable!

Details

- Use Selenium WebDriver (aka Selenium 2.0): http://docs.seleniumhq.org/projects/webdriver/
- Implement the PageObject pattern: https://github.com/SeleniumHQ/selenium/wiki/PageObjects - or tell us why your alternative approach is better.
- Write your test in any language WebDriver supports.
- Sign up for a free trial of Confluence Cloud at https://www.atlassian.com/software/confluence/try/ . This will provide you with 7 days of free access to a hosted Confluence instance (no credit card is required).

Submitting the Results

Once you've written your tests, run them successfully and are happy with the result:

- Push your tests to your Bitbucket repository.
- Document any test pre-requisites, assumptions made and any issues found in the repo's ReadMe file.

Exercise 2 - Exploratory Testing

Goal

To demonstrate your ability to identify risks in a feature without using specifications or scripted test cases.

Task

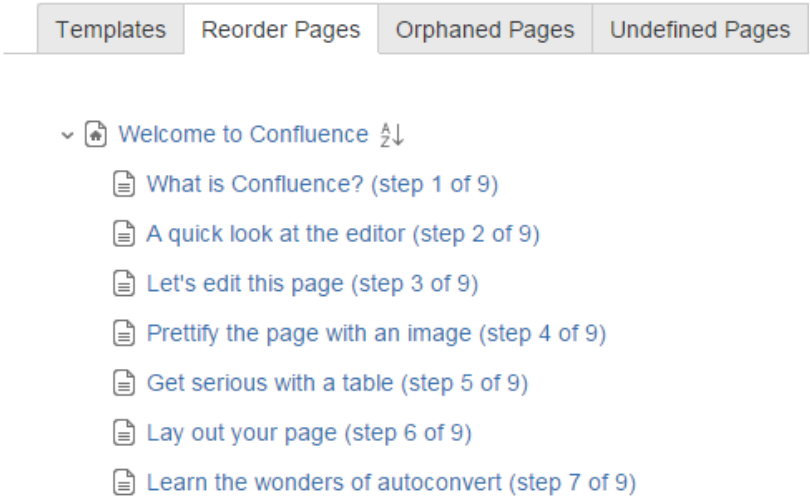
A graduate developer has written a new feature in Confluence called Reorder Pages. They have already tested that basic happy-path scenarios for sorting, dragging and dropping are working. They are now asking for your help to determine whether the feature is ready for release to production.

Perform 30 minutes of exploratory testing of the feature to investigate if it is of high quality.

After you complete your testing, determine what further testing the feature may need before you would consider it ready to ship.

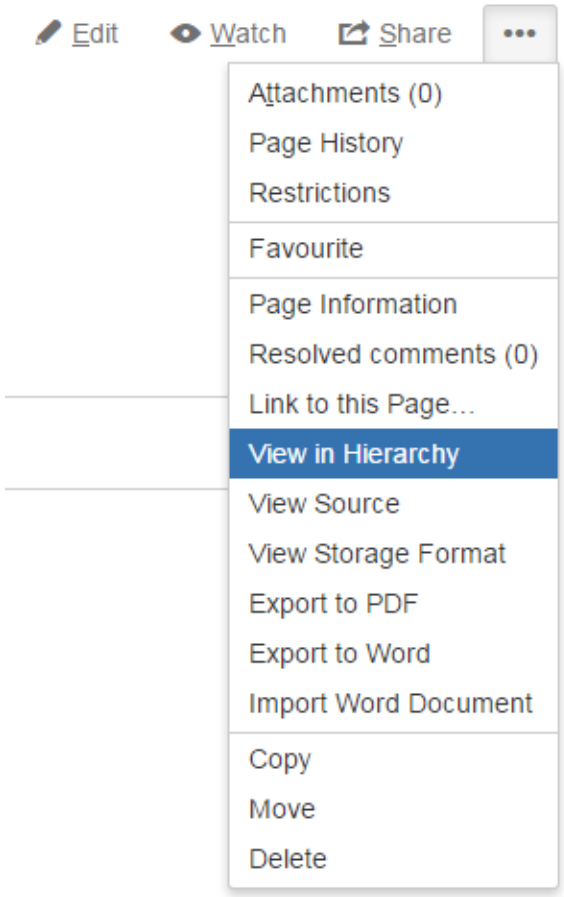
Details

- You can use the same Confluence instance that you created for the previous exercise.
- You will be testing the Page Tree. This feature displays all the pages in a space in an expandable and collapsible tree. It allows the user to rearrange these pages by drag-dropping.



An example page tree - your data may be different

- This feature can be accessed by clicking “View in Hierarchy” from the “...” menu in the top-right corner on any page in a space.



How to access the Page Tree from any Confluence page

- The scope of this exercise is limited to the tree view feature only, under the “Reorder Pages” tab. You should not test the other tabs (Orphaned Pages, Undefined Pages, etc.) or other Confluence features, unless they will help you to assess the quality of this feature.
- You should explore to discover other functionality offered by this feature.
- As this exercise is aimed at assessing your exploratory testing approach and not the total number of bugs found, there is little value in continuing beyond the allocated 30 minutes.
- Aim to go beyond basic happy-path functional testing, and consider other areas of risk.
- You can spend as much time as you need coming up with further testing areas and writing up your results.

Submitting the Results

After completing your exploratory test session, create a file containing:

- A short description (1-2 sentences) of the approach you took to your testing.
- A bulleted list of the scenarios you tried, even if they were successful. For example:
    - “Tried to drag a parent page to become a grandchild of itself”
    - “Tried to use the web service to make a page a child of itself”
- A quick description of any bugs you found. It’s OK to not have found any – we’re more interested in your test ideas and approach than what you executed in the 30 minutes.
- A list of further testing areas that you would want completed before you would be comfortable shipping this feature to millions of users. Consider the risks of the feature as you understand it.

Once completed, please push the file to your Bitbucket repository.