# Technical Web Test

**Question 1**

Introduction:

The following is the main form of an application that converts values from one currency to another. The collection of currencies, exchange rates, and rules for applying commission are held in a shared database, accessed by multiple users of this application on a network. The user of this form selects a base currency, a target currency, specifies an amount (expressed in the base currency), can choose to include or exclude commission, then hits the Convert button to generate the equivalent value in the target currency. That value is shown where “N/A” is displayed in the screenshot below.

The separate “Maintain currencies” form (not shown) provides operations for adding, deleting and editing currencies. A separate process constantly updates the database with exchange rates for any recognised currencies (you do not need to worry about how this works, but you do have control over stopping and starting it).

The separate “Maintain commission rates” form (not shown) provides operations for specifying a minimum commission value and a collection of commission rates for different ranges of values. The minimum commission value and ranges are defined in Sterling. For example, for “amount to convert” values in the range 0.01 to 100.00 pounds Sterling, the commission rate is 5%. An example minimum commission value is £10.



In the following questions identifying tests is adequate, details of test data and expected results are not required where they can be inferred. Aim for an answer no longer than 1200 words for this entire section:

1. Using the database and the GUI, how would you test the functionality of this form?

**I would use the database to identify the values at which the exchange rate changes and use this data to drive the tests ensuring that the boundaries and standard inputs are working as expected.**

**I would identify a set of base and target currency scenarios to test against to ensure this in conjunction with the amount to convert is as expected.**

**I would also check that when commission is included in the calculation that it is applied correctly.**

**I would perform a set of UI tests providing invalid/no data into various fields to ensure error prompts are correctly displayed and are appropriate for the error.**

(b) What tests would you perform to test the operations supported by the “Maintain currencies” form?

**I would perform the addition, editing, and deletion of currencies to ensure this is working correctly.**

**I would verify that when a new currency is added which is a recognised currency then it is automatically updated by the separate process mentioned.**

**I would check that if an exchange rate is automatically updated while the user is editing the associated currency then they are informed of this and provided options on how to proceed.**

**I would check that the automatic update process can be turned off/on.**

(c) What tests would you perform to test the operations supported by the “Maintain commission rates” form?

**For all inputs on this form I would verify that only valid data can be entered (Only positive numerical values within appropriate ranges).**

**Check that the higher amount of either the minimum commission value or the percentage commission rate specified is taken depending on values entered.**

**Check that commission rates for different ranges of values are calculated correctly including standard and boundary testing.**

(d) What tests would you perform on the form above in order to test accessibility?

**Check that user can change text font and size for improved readability.**

**Ensure a high contrast beteen the colours of all text and its associated background.**

**Check that screen reader (text to speech) software works on the form.**

**Check that the user can navigate and utilise the form using the keyboard (such as tabbing through fields or having jump to field alt-commands).**

(e) If you were automating the testing of this form for regression test purposes, what would you hope the developer had done that would make the automation easier?

**To make automation easier the developers can add IDs to all controls (Such as the ‘convert’ button), input fields (Such as the ‘Base Currency’ dropdown), and labels/non-editable fields that show any output (Such as the ‘amount in target currency’).**

(f) If this application were used internationally, what further tests would you perform?

**I would perform tests to check that the product can be easily reconfigured for other languages (if required), and that the base currency and any other values are configured for their locale.**

**I would additionally perform tests to ensure that the conversion mechanism when converting from a currency other than pounds sterling to any other currency still performs as expected and the right result is returned (Including tests for commission rates).**

**This could be done on the UI to check the conversion mechanism has been implemented correctly and values are able to be correctly displayed along with a set of associated API tests against the backend to check that the actual conversion mechanism behaves as expected when using alternative currencies.**

(g) Looking at the screenshot of the form above, what user interface issues are immediately apparent (you may have mentioned some of these in previous answers)?

**The fields ‘base currency’ and ‘target currency’ are short in length which might mean some currencies are not fully readable when the dropdown is displayed. Given the length of the ‘Amount to convert field’ I would make these two fields longer.**

**No supportive text is provided to tell the user how to represent the value they enter in the ‘Amount to convert’ field (i.e. whole pounds only, pounds/pence).**

**Question 2**

For the following technical test, you can use any JavaScript framework of your choice. We want you to automate the scenarios below using Chrome browser.

1. Open the following URL: <http://www.willistowerswatson.com/ICT>
2. Change the language and region from top left corner to United States English, see example snippet.



1. Search for the word “IFRS 17” using the search box
2. Validate if you have arrived on the result page
3. Check if the result is sorted by “Date”. If not, sort by “Date”
4. Use the “Filter by” functionality and set Content Type to “Article”
5. Validate that each article in the list displays a link that starts with “https://www.willistowerswatson.com/en-US/”

**Evaluation** will be done on the following criteria:

* Clean and tidiness of the solution
* Design and architecture
* Readability
* Stability of the tests

Please provide instructions for setting up the environment and how to run the tests. Share your code either in a public Git repo or compress the files and mail them back.

**This project has been built in Java Eclipse using Selenium, Cucumber, and Junit.**

**The dependencies are managed by Maven. You will need to install Eclipse, the Cucumber plugin, and open this project as a Maven project.**

**The Chrome driver provided in the project is for Chrome browser version ‘75.0.3770.142’. If using a different version of chrome please go to this website and download the corresponding driver:** [**http://chromedriver.chromium.org/downloads**](http://chromedriver.chromium.org/downloads)

**To execute the tests right click on the ‘TestRunner.java’ and select Run as Junit Test.**

**The results are displayed both in the console and in the ‘cucumber-reports’ folder in the target directory.**