**Automation Technical Test - Web Applications**

**Purpose**

As part of our hiring process for technical roles, we require all candidates to complete a small technical exercise. This is an integral part of our process to assess an individual’s technical ability when completing an activity in their own time.

Given the importance of this exercise to our hiring process, we highly recommend you give this adequate consideration and address this task as you would do any other professional assignment in your current/previous workplace.

**Instructions**

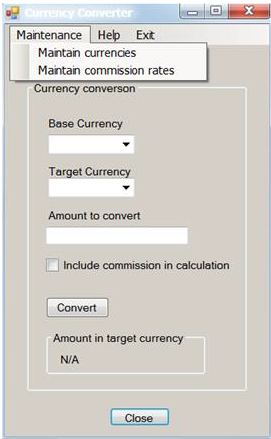
* There are two parts to this technical assessment
  + A written assessment of your ability to tackle a typical testing problem you might face as a test analyst
  + A code-based assessment, testing your ability to automate a test scenario
* Written answers for part 1 should be submitted in a suitable format - e.g. a word document
* Please work on a GitHub, create the New branch within this repository and when you are happy to submit, create a pull request back into the main branch
* Please use the pull request to comment on any aspects of your solution that you didn’t have time to complete, are not complete to your satisfaction or are not working
* Would you please add a markdown file with instructions on how to set up and run your solution to question 2

**Question 1**

**Introduction**

The following is the main form of an application that converts values from one currency to another. The application has the following characteristics:

* 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. The screenshot below shows that value where “N/A” is displayed.
* 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.



Given the information provided, please answer the following questions:

1. Using the database and the GUI, how would you test the functionality of this form?
2. What tests would you perform to test the operations supported by the “Maintain currencies” form?
3. What tests would you perform to test the operations supported by the “Maintain commission rates” form?
4. What tests would you perform on the form above in order to test accessibility?
5. 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?
6. If this application were used internationally, what further tests would you perform?
7. The screenshot of the form above shows immediately apparent user interface issues (you may have mentioned some of these in previous answers)?

**Notes**

* In answering the questions, identifying tests is sufficient
* 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

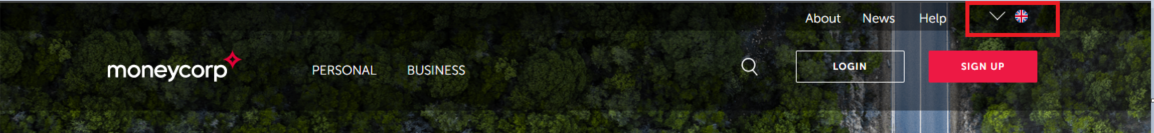
**Question 2**

**Introduction**

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

**Test Scenario**

1. Open the following URL: <https://www.moneycorp.com/en-gb/> and verify
2. Change the language and region from the top right corner to USA (English) and verify See example snippet.



1. Click Find out more for “Foreign exchange solutions” Validate if you have arrived on the page
2. Search for the word “international payments” using the search box
3. Validate if you have arrived on the result page
4. Validate that each article in the list displays a link that starts with <https://www.moneycorp.com/en-us/>

**Evaluation**

Evaluation will be done on the following criteria:

* Clean and tidiness of the solution
* Design and architecture
* Readability
* Maintainability
* Stability of the tests
* Clarity of setup and execution documentation

Would you please provide instructions for setting up an environment from scratch and how to run the tests?

#### **David Bailey**

#### **21st March 2022**

**Question 1**

1. Using the database and the GUI, how would you test the functionality of this form?
   1. Consider requirements
      1. Determine possible values of the base and target currencies
      2. Determine possible commission rules
      3. User groups
      4. Non-functional requirements
   2. Identify test data set
   3. Request typical examples (example-based testing)
   4. Automate to achieve a large, repeatable set of tests
   5. Compare database values with observed output, including:
      1. Menu options
      2. Commission options
      3. Calculation outputs
2. What tests would you perform to test the operations supported by the “Maintain currencies” form?
   1. Black-box / functional scenarios
      1. Happy paths
      2. Error scenarios
         1. Negative values
         2. Missing values (e..g not specifying a target currency)
         3. Default values
         4. Too large values
         5. Attempt to achieve negative output
         6. Are errors gracefully handled and to specification?
         7. Undefined commission rates
         8. Extermely different Base-Target vs. Target-Base
         9. Database or network unavailable
      3. Equivalence partioning, e.g. ranges for commission rates
      4. Boundary value analysis (e.g. around and on commission rate changes)
      5. Rounding rules (bankers rounding, nearest neighbour)
      6. Compare database values to observed application behaviour
      7. Floating-point calculation errors (e.g. 0.1+0.2 in Javascript)
      8. Minimum commission rates
      9. Different user groups, e.g. administrators for exchange rates and commission rates, end-user.
      10. Usability, e.g. clicking on fields, copy+paste, using the menu
   2. Non-functional
      1. Performance aspects of the application, e.g. speed of calculation, speed of identifying database changes
      2. Database logging & UI logging/debugging
      3. Load testing
      4. Security
      5. Installation method
      6. Documentation
      7. Dependencies
      8. Memory leaks
      9. Use of thousand separators, e.g. comma, dot, short-space
      10. Operating systems and versions
      11. Browser compatibility
      12. Device compatibility
      13. Upgrade process for new releases
   3. Legal considerations, e.g. levels of testing required for financial accreditation
   4. Unusual values, e.g. currencies with no decimals, currencies with extreme conversion rates, e.g. Base:A🡪Target:B very different from Base:B🡪Target:A
   5. Display of characters such as currency symbols for internationalisation
3. What tests would you perform to test the operations supported by the “Maintain commission rates” form?
   1. Ability to reach the form
   2. Ability to view accurate list of base and target currencies
   3. Ability to input reasonable values for commission rates
   4. Ability to input a minimum commission rate
   5. Ability to input ranges of commission rates, without gaps between ranges
   6. Ability to handle invalid inputs gracefully
4. What tests would you perform on the form above in order to test accessibility?
   1. Font size, style and contrast with the background
   2. Screen-reader compatibility
   3. Achieving usability standards
   4. Ability to tab between the menu and fields, e.g. keyboard-only usage
   5. Tabbing fields in a logical order
   6. Changing appearance with the operating system’s settings, e.g. high contrast
   7. Large fonts still fitting in the forms and fields
   8. Fonts correctly handling currency, number and separator symbols
   9. Help text and hover text on fields
5. 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?
   1. Provide API or access to the test database
   2. Provide unique and logical names to the fields
   3. Provide a method to record the version and patch level of the software
   4. Considered non-functional aspects of the solution during development
   5. Already covered unit testing
   6. Documented the solution and code
   7. Provide features for debugging while testing, e.g. logging network calls, error codes, crash dumps
6. If this application were used internationally, what further tests would you perform?
   1. Response to locale settings, including separators, timezones, languages, symbols
   2. Ability to configure these settings within the solution (e.g. choose decimal separator preferences)
   3. Provide user documentation and error messages in appropriate languages
   4. Consider replication of databases to local instances for access and performance reasons
   5. Software distribution and upgrade methods
7. The screenshot of the form above shows immediately apparent user interface issues (you may have mentioned some of these in previous answers)?
   1. “Currency converson” contains a typo
   2. Title bar is low-contrast
   3. Consider separating the ability to administer exchange rates and commission rates from end-users.
   4. Consider using a browser interface for compatibility, accessibility and avoiding software distribution, and quality of appearance
   5. Consider the overall purpose of the software, should the user then be able to request the conversion? Should the user be shown the breakdown of the calculation (e.g. amount of commission)?
   6. Should the user have to enter the decimal point themselves?
   7. Consider using flags as a visual aid for each currency for ease-of-use.
   8. Exit menu option, Close button and X on the window all have the same function.
   9. The “Convert” button is superfluous, the software could dynamically update the result

**Question 2**

**Code and documentation can be found at:**

<https://github.com/davidatbailey/AutomationExercise1>

Test **navigation\_001.spec.js** covers:

* Open the following URL: <https://www.moneycorp.com/en-gb/> and verify
* Change the language and region from the top right corner to USA (English) and verify See example snippet.
* Click Find out more for “Foreign exchange solutions” Validate if you have arrived on the page
* Search for the word “international payments” using the search box
* Validate if you have arrived on the result page
* Validate that each article in the list displays a link that starts with <https://www.moneycorp.com/en-us/>

Would you please provide instructions for setting up an environment from scratch and how to run the tests?

* See README.md for set-up and execution instructions