Manual Test Summary

API Testing

Feature: Create Booking

Scenario: Create a booking using valid details
When I create a booking using valid details
Then a 200 response code is returned
And the booking details are returned

Result: Passed

Scenario: Create a booking using a numeric character for the first name
When I create a booking using a numeric character for the first name
Then a 400 response code is returned

Result: Failed - 200 response code is returned and the booking is created

Scenario: Create a booking using an empty string for the first name
When I create a booking using an empty string for the first name
Then a 400 response code is returned

Result: Failed - 500 response code is returned

Scenario: Create a booking using a numeric character for the first name
When I create a booking using a numeric character for the first name
Then a 400 response code is returned

Result: Failed - 200 response code is returned and the booking is created

Scenario: Create a booking using an empty string for the last name
When I create a booking using an empty string for the last name
Then a 400 response code is returned

Result: Failed - 500 response code is returned

Scenario: Create a booking using a monetary value that has 1 decimal place for the price
When I create a booking using a monetary value that has 1 decimal place for the price
Then a 200 response code is returned
And the price is formatted to 2 decimal places

Result: Failed - 200 response code is returned but the price is stored to 1 decimal place

Scenario: Create a booking using a monetary value that has 3 decimal places for the price
When I create a booking using a monetary value that has 3 decimal places for the price
Then a 400 response code is returned

Result: Failed - 200 response code is returned and the price is stored to 3 decimal places

Scenario: Create a booking using an alpha character for the price
When I create a booking using an alpha character for the price
Then a 400 response code is returned

Result: Failed - 500 response code is returned

Scenario: Create a booking using an empty string for the price
When I create a booking using an empty string for the price
Then a 400 response code is returned

Result: Failed - 500 response code is returned

Scenario: Create a booking using a non boolean value for the deposit
When I create a booking using a non boolean value for the deposit
Then a 400 response code is returned

Result: Failed - 200 response is returned and the deposit is set to true

Scenario: Create a booking using an empty string for the deposit
When I create a booking using an empty string for the deposit
Then a 400 response code is returned

Result: Failed - 200 response is returned and the deposit is set to false

Scenario: Create a booking using a numeric character for the check in date
When I create a booking using a numeric character for the check in date
Then a 400 response code is returned

Result: Failed - 200 response is returned and the check in date is set to 01-01-1970

Scenario: Create a booking using an alpha character for the check in date
When I create a booking using an alpha character for the check in date
Then a 400 response code is returned

Result: Failed - 500 response code is returned

Scenario: Create a booking using an empty string for the check in date
When I create a booking using an empty string for the check in date
Then a 400 response code is returned

Result: Failed - 500 response code is returned

Scenario: Create a booking using a numeric character for the check out date
When I create a booking using a numeric character for the check out date
Then a 400 response code is returned

Result: Failed - 200 response is returned and the check out date is set to 01-01-1970

Scenario: Create a booking using an alpha character for the check out date
When I create a booking using an alpha character for the check out date
Then a 400 response code is returned

Result: Failed - 500 response code is returned

Scenario: Create a booking using an empty string for the check out date
When I create a booking using an empty string for the check out date
Then a 400 response code is returned

Result: Failed - 500 response code is returned

Scenario: Create a booking using a check in date that is later than the check out date
When I create a booking using a check in date that is later than the checkout out date
Then a 400 response code is returned

Result: Failed - 200 response code is returned and the booking is created

Feature: Get All Bookings

Scenario: Get all bookings
Given multiple bookings exist
When I get all bookings
Then a 200 response code is returned
And a list of all bookings is returned

Result: Passed

Feature: Get Booking

Scenario: Get an existing booking
Given a booking exists
When I get an existing booking
Then a 200 response code is returned
And the booking details are returned

Result: Passed

Scenario: Get a non existent booking
When I get a non existent booking
Then a 404 response code is returned

Result: Passed

Scenario: Get a delete booking
Given a booking exists
And I have deleted the booking
When I get deleted booking
Then a 404 response code is returned

Result: Passed

Feature: Delete Booking

Scenario: Delete an existing booking with auth
Given a booking exists
When I delete an existing booking with auth
Then a 201 response code is returned

Result: Passed

Scenario: Delete an existing booking without auth
Given a booking exists
When I delete an existing booking without auth
Then a 403 response code is returned

Result: Passed

Scenario: Delete a non existent booking
Given a booking exists
When I delete a non existent booking

Then a 404 response code is returned

Result: Failed - 405 response code is returned

UI Functional Testing

Feature: Create Booking

Scenario: Create a booking using valid details
When I create a booking using valid details
Then booking is created successfully

Result: Passed

Scenario: Create a booking without entering any details
When I create a booking without entering any details
Then booking is creation is unsuccessful
And an error message is displayed

Result: Failed - The booking is not created but the user does not receive an error

Scenario: Create a booking on a separate tab
Given I have the list of bookings open on 2 tabs
When I create a booking using valid details on tab 1
And I view tab 2
Then the booking created from tab 1 is displayed

Result: Failed - The booking is not displayed until the page is refreshed

Feature: View Bookings

Scenario: View all bookings
Given a booking exists
When I view all bookings
Then the existing bookings are displayed

Result: Passed

Feature: Delete Booking

Scenario: Delete an existing booking
Given a booking exists
When I delete the booking
Then booking is deleted successfully

Result: Passed

Scenario: Delete a booking that has already been deleted
Given a booking exists
And I have the list of bookings open on 2 tabs
When I delete the booking on tab 1
And I view tab 2
And I delete the same booking in tab 2
Then an error message is displayed as the booking has already been deleted

Result: Failed - The user does not receive an error

UI Usability Testing

The following are examples of issues found when performing usability testing on the application:

- Use of tools such as WAVE which can be used for accessibility testing (https://wave.webaim.org/report#/http://hotel-test.equalexperts.io/) and highlight accessibility issues. Some examples of these issues are; missing labels on fields, inconsistent page structure and missing page regions.
- Formatting and colour choices of heading and booking table could be improved to make the page more readable.
- Pagination/booking search could be added to the booking table to help with the users experience when large amounts of data are present.
- The booking table sometimes renders incorrectly and rows are not aligned as expected.
- Fields in the database are not limited on characters and cause columns to overlap if the field contains a a word with many characters (e.g. a first name containing 30 characters will overlap into the surname column).

Typically most of these issues would be prevented during the UI/UX stage of the product life cycle.

Performance Testing

Performance testing should be performed to ensure the application meets the benchmark for user requests. Some of examples of tests that can be carried out are as follows:

- Load testing the system to see how the application performs under high loads by monitoring response times. These loads can be manipulated by modifying test parameters such as requests per second, concurrent users etc. This type of testing may identify issues with the application such as memory leaks.
- Stress testing could be used to identify how the application performs under loads that may push the system past its limit and observe how the system recovers from failure.

Security Testing

Security testing should be performed to ensure the application is protected against any vulnerabilities that may pose a threat to the user, application and the system. Vulnerabilities in the application may result in personal information being captured by a third party, or a systems database being manipulated without permission, amongst many other potential threats

Some examples of security testing that should be carried out on the application are as follows:

- XSS (Cross Site Scripting) testing should be performed on the input fields and the URL to ensure that it is not possible to modify a users experience or steal website data such as session information.
- SQL injection testing should be performed on input fields and the URL to prevent the application database from being manipulated or exposing any data that should not be known as an end user.

Notes:

- The connection to the application is insecure and has an invalid SSL certificate
- The first name and surname fields allow for XSS attacks as they don't strip tags e.g. <script> e.g. <script>window.location='https://www.equalexperts.com'</script> in the first name field would redirect the user to https://www.equalexperts.com/ when the table loads

Cross Browser Testing

Cross browser testing should be performed on the application to ensure the application in a way that can be viewed across multiple different browsers, version and operating systems. This list of required browsers would usually come from requirements. I have performed testing against the following 3 browsers and found the following issues:

macOS Catalina Version 10.15.2 Google Chrome Version 79.0:

- Booking table rows do not render correctly

macOS Catalina Version 10.15.2 Safari Version 13.0.4:

- Booking table rows do not render correctly
- Booking table displays multiple lines which disappear once highlighted
- Deposit dropdown changes style on page zoom in/out

iOS Version 13.3 Safari Version 13:

- Booking table does not render correctly
- Focus of page does not change when new row is added