# Test Plan Document

## *Challenging DOM*

*Shashank Sahu*

*Version 1.0*

*18th August 2021*

# 1. Introduction

This document describes how the tests on the Challenging DOM web page are to be planned and executed.

# 2. Scope

**Feature testing** of web page available at below route:

**http://the-internet.herokuapp.com/challenging_dom**

***In Scope:***

- The browser to be used is the latest version of Chrome.
- Simple Features testing only
- Automated Tests

***Out of Scope:***

- Other browsers out of scope.
- Cross Browser Testing.
- Excludes scalability and resilience tests.

# 3. Test Strategy

The test strategy here is to do spot checks on various parts of the web page, including elements such as buttons, table elements and it's content, canvas and links.

# 4. Environment Requirements

- All the dependencies of libraries used while writing the automated Tests
- A PC running Selenium and the latest version of Chrome browser with the corresponding chromedriver application.
- Java is required for Selenium to run.
- Good internet connection.

# 5. Test Schedule

Tests to be performed during the discussion on Monday 23rd August 2021.

# 6. Functions To Be Tested

- Page title
- Buttons
- Table elements
- Hyperlinks

## 7. Resources and Responsibilities

*Shashank Sahu* will be the creator of documentation and tests.

## 8. Deliverables

- This Test Plan
- Automated Test code

## 9. Entry Criteria

- Test Builds available.
- AUT build passes the Smoke Tests.

## 10. Exit Criteria

- All automated tests running without fail.
- No Show Stopper bugs at the time of Go-NoGo call.
- Priority 2 & severity 2 bugs less than 3% of all the open bugs.

## 11. Dependencies

- Web page under test being available and working properly
- Good connection from the host computer to the website

## 12. Risks

- Web page under test is modified during tests, rendering results invalid.
- No version control is available so that results can be matched to the target revision of the web page under test.

## 13. Mitigation

- Test Reports of previous test runs can be used to show the test results of AUT.

## 14. Tools

Tools used include:

- IntelliJ IDE.
- Selenium for UI automation.
- Chrome browser.
- Maven build tool.
- Jenkins setup in Local for CI-CD.
- Cucumber/gherkin for feature writing.

## 15. Documentation

All documentation is provided in-line with the test code

## 16. Approvals

Tests and documentation will be reviewed during the interview on 23rd August 2021.

## 17. Source Code

The Automation framework along with the automated tests can be found at below git location :
***https://github.com/sahuShashank12/CucumberHybridFramework***

## 18. Test Cases

```
Feature: Tests Challenging DOM Page

 @Regression
 Scenario: User is able to load the Challenging DOM Page
   When user hits the url in browser
   Then page loads successfully

 @Regression
 Scenario: Page Title Details
   Given user is on the dom page
   When user gets the title of the page
   Then page title should be "The Internet"

 @Regression
 Scenario: Page Header Details
   Given user is on the dom page
   Then page Header should be "Challenging DOM"

 @Regression
 Scenario: Page Description Details
   Given user is on the dom page
   Then page Description should be "The hardest part in automated web testing is finding the best locators (e.g., ones that well named, unique, and unlikely to change). It's more often than not that the application you're testing was not built with this concept in mind. This example demonstrates that with unique IDs, a table with no helpful locators, and a canvas element."

 @Regression
 Scenario: User is able to click on blue button and answer in canvas updates
   Given user is on the dom page
   And user fetch value from canvas
   When user clicks blue button
   Then the answer canvas updates
```

```
@Regression
Scenario: User is able to click on red button and answer in canvas updates
  Given user is on the dom page
  And user fetch value from canvas
  When user clicks red button
  Then the answer canvas updates

@Regression
Scenario: User is able to click on green button and answer in canvas updates
  Given user is on the dom page
  And user fetch value from canvas
  When user clicks green button
  Then the answer canvas updates

@Regression
Scenario: User is able to interact with Canvas answer field
  When user is on the dom page
  Then answer canvas is visible

@Regression
Scenario: User is able to see the correct table headers
  Given user is on the dom page
  When user fetches table headers
  Then the table headers should be
  |Lorem|Ipsum|Dolor|Sit|Amet|Diceret|Action|

@Regression
Scenario: User is able to get correct number of table rows
  Given user is on the dom page
  When user fetches table rows
  Then number of rows should be 10

@Regression
Scenario: User is able to get correct number of table columns
  Given user is on the dom page
  When user fetches table columns
  Then number of columns should be 7

  ## As unable to Edit row on Edit Action Verifying the action with URL change
@Functional
Scenario: User is able to edit first row
  Given user is on the dom page
  When user fetches details of row 1
  And clicks on edit button
  Then action should be performed "#edit"
```

```
  ## As unable to Delete row on delete Action Verifying the action with URL change
@Functional
Scenario: User is able to delete second row
  Given user is on the dom page
  When user fetches details of row 2
  And clicks on delete button
  Then action should be performed "#delete"


  ## Deliberately failing the Test to Capture Failed scenario in Extent Report
@Functional
Scenario: Footer link loads
  Given user is on the dom page
  Then Footer link text should be "Elemental Selenium.."
```