# CX Playwright POC

## Purpose

### The purpose of this Proof of Concept (POC) is to evaluate the viability and efficiency of Playwright for automating web application testing within the CX project.

### This POC demonstrates the basic test automation capabilities, including validation of page elements and content, using Playwright.

## Pre-requisite

#### Node.js

### Node.js is a JavaScript runtime that is required to run Playwright and its dependencies.

### By default, it also installs npm (Node Package Manager), which will be used to install Playwright and other packages.

#### Visual Studio Code (VS Code) Editor

### Visual Studio Code is a popular code editor that will be used for writing and managing the Playwright test scripts.

#### Install Playwright

### Playwright is the framework used for browser automation in this POC.

## Test Cases

##### Test Case 1: TC7510337\_GPTI\_Portal\_Open\_A\_General\_Help\_Case\_And\_Validate

### The goal of this test case is to ensure that the customer care section of the portal is functioning correctly after a successful login.

### This includes validating the visibility of the title, the correct count and values of the dropdown options, the accuracy of the table header values, and verifying that the most recent date and time appear first in the list.

**Test Steps:**

#### Login to the Portal:

### Navigate to the login page of the portal.

### Enter valid login credentials (username and password).

### Click the login button to access the user dashboard.

#### Validate Customer Care Title Visibility:

### After logging in, navigate to the customer care section of the portal.

### Ensure that the title of the customer care section is visible and correctly displayed.

#### Validate Dropdown Options:

### Locate the dropdown menu within the customer care section.

### Verify that the dropdown contains the correct number of options.

### Validate that each dropdown option has the expected value.

#### Validate Table Header Values:

### Identify the table displayed in the customer care section.

### Verify that the table headers are correct and correspond to the appropriate column data.

#### Validate Date and Time Order:

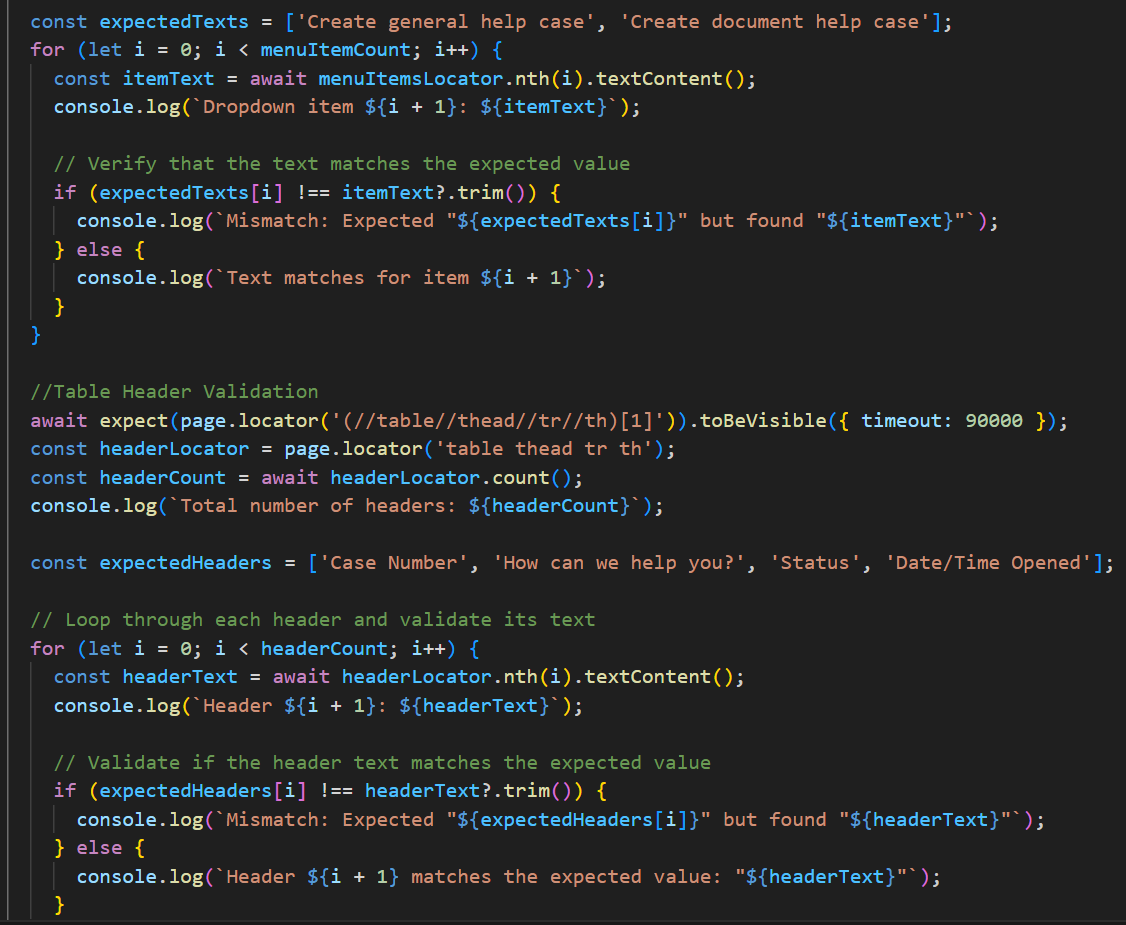
### Ensure that the date and time listed in the customer care table are accurate.

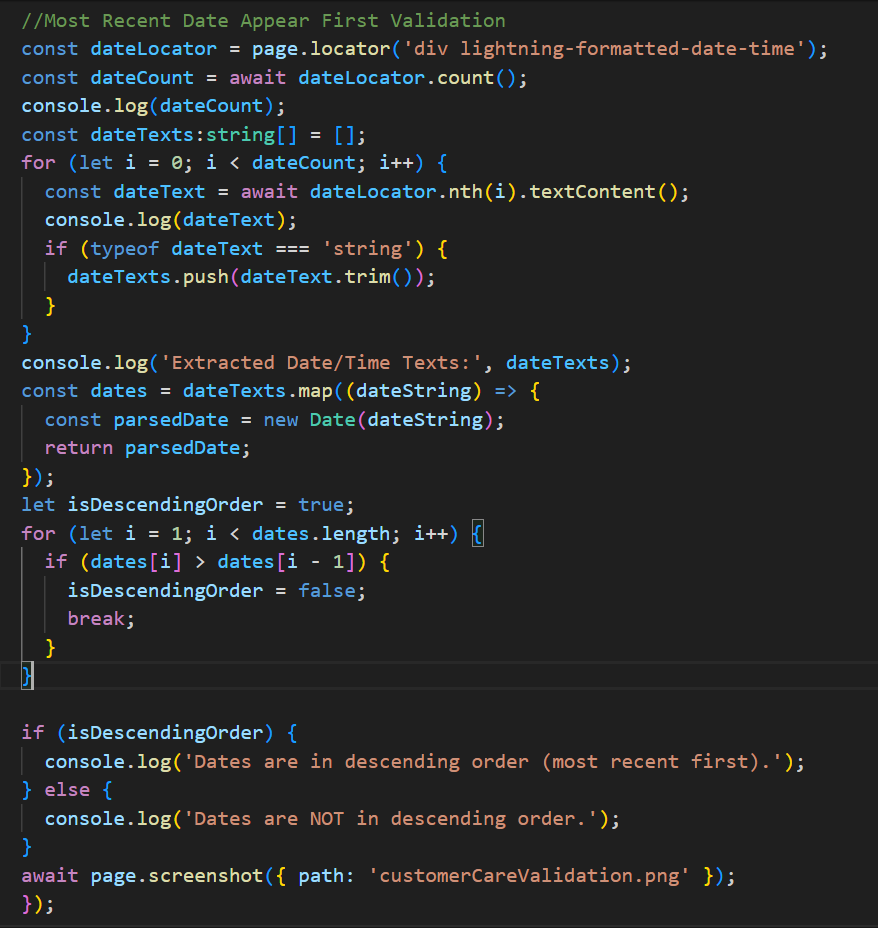
### Verify that the most recent date and time appear first in the list, and the dates are in descending order.

##### Playwright Automation Test Script

A screen shot of a computer program

Description automatically generated



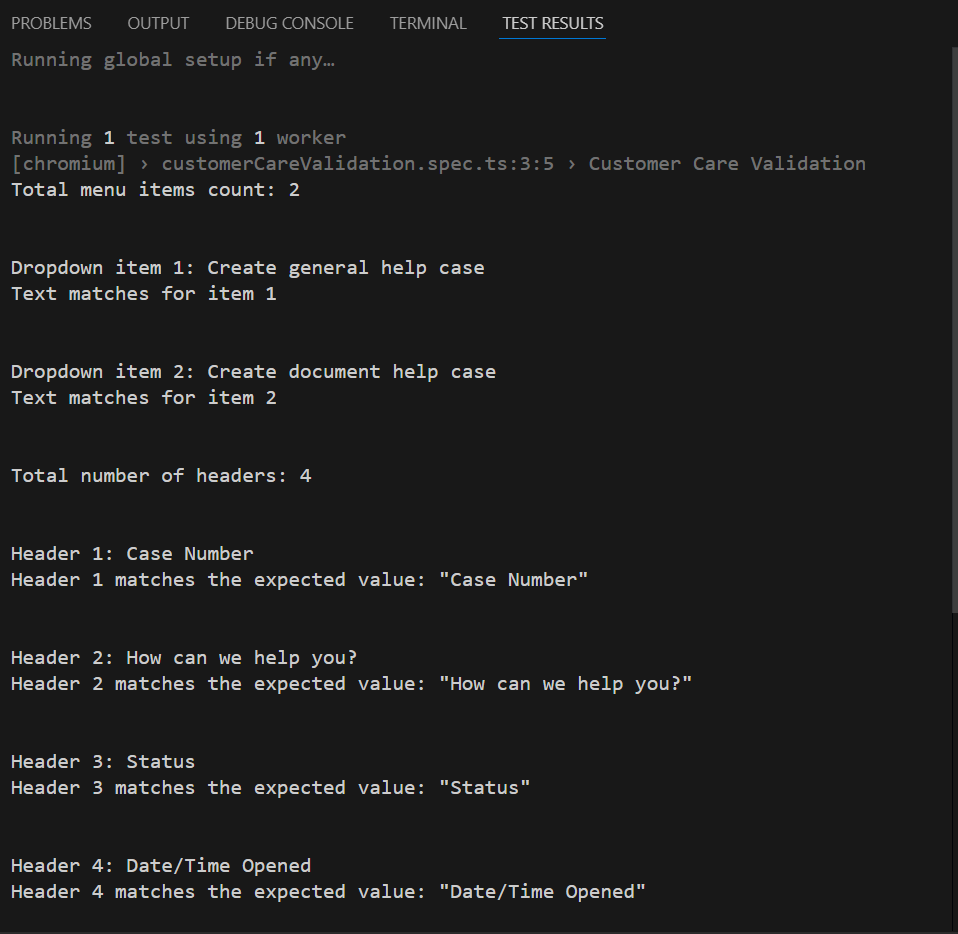


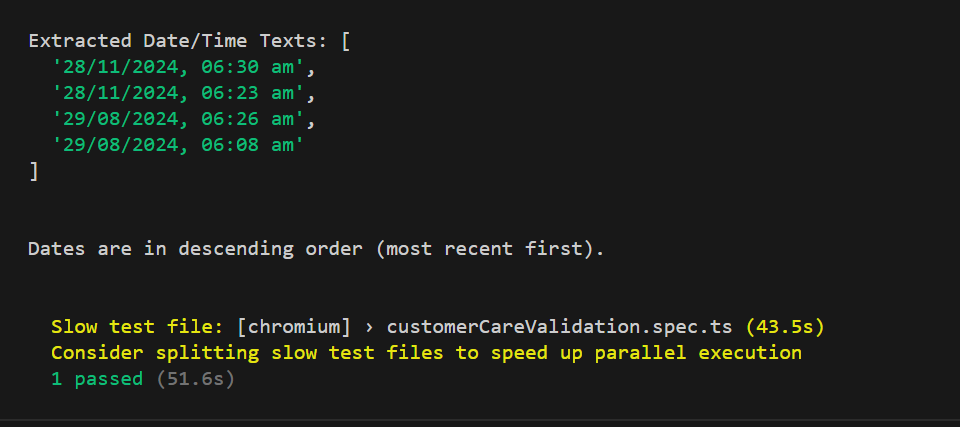
##### Expected Outcome:

### Playwright will provide detailed feedback in the terminal regarding the success or failure of the test cases.

### We will see output indicating whether the tests passed or failed, and detailed logs will be provided for any failed tests, including screenshots, console logs, and trace files.







##### Screenshot

A screenshot of a computer

Description automatically generated

##### Test Case 2: TC7510385\_GPTI\_Portal\_Request\_To\_Add\_Another\_Account\_To\_My\_Permissions

### The objective of this test case is to automate the process of creating a new account, requesting access to specific features, and generating a case.

### The test will validate that a new account is created successfully with valid details, the requested features are correctly added to the account, a case is generated and associated with the account, the BP entitlement number is generated successfully for the case.

##### Test Steps

#### **Login to the Portal**:

### Navigate to the login page of the portal.

### Enter valid login credentials (username and password).

### Click the login button to access the user dashboard.

#### Create a New Account:

### Navigate to the profile settings section of the portal.

### Enter valid account number and select the features to be requested for the new account.

### Verify that the new case is successfully created, and the case details are displayed correctly in the case list in salesforce.

#### Validate Case Details:

### Navigate to the created case.

### Verify that the case contains the correct details, such as case owner, case origin, case number, request type, requested features, and any other relevant data.

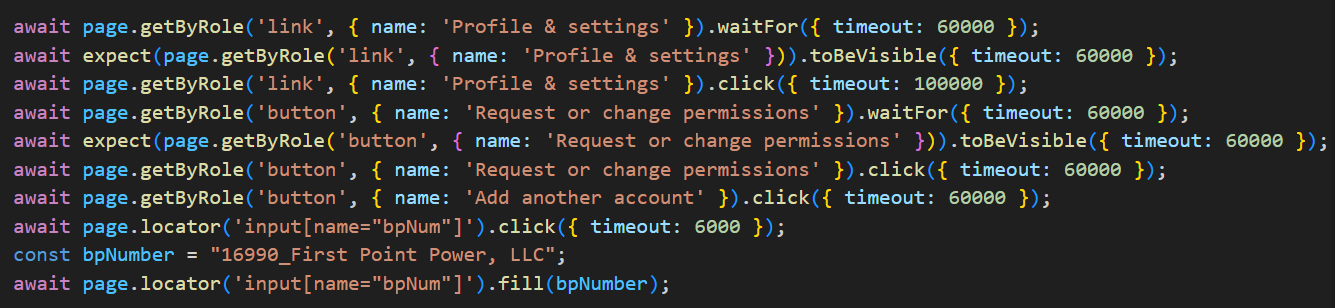
### Select the BP account number in Requested BP field and approve the case.

#### Validate BP Entitlement Number Generation:

### Ensure that after case creation and approval, the BP entitlement number is generated.

### Verify that the entitlement number is unique and correctly associated with the created case.

##### Playwright Automation Test Script



A screen shot of a computer code

Description automatically generated

A computer screen shot of text

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

##### Expected Outcome

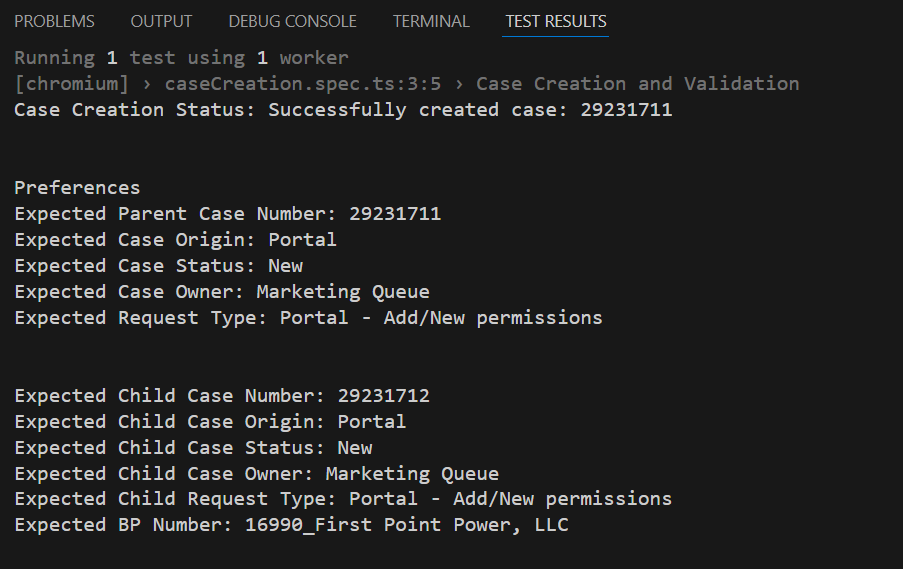


##### Screenshot

A screenshot of a computer

Description automatically generated

##### Terminal Outcome

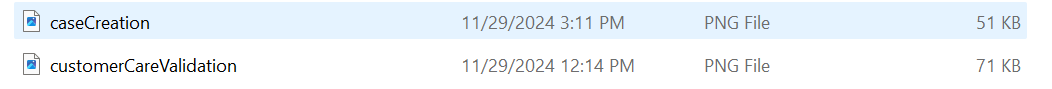


A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated



## Test Case Execution Times

### The following table summarizes the **execution times** of key test cases

|  |  |  |
| --- | --- | --- |
| **Scenario Type** | **Test Case Name** | **Total Execution Time** |
| Medium | TC7510337- Customer Care Validation | 51.6s |
| Complex | TC7510385 - Case Creation and Validation | 1.4m |

## Test Case Design Effort Estimation

|  |  |  |
| --- | --- | --- |
| **Task/Activity** | **Estimated Time (Hours)** | **Description** |
| **Understand Application Flow** | 1 hour | Analyze the application's overall flow and functionality to gain a thorough understanding of its behavior. |
| **Design Test Case 1(Medium)** | 5 to 6 hours | Design test case 1 (logic implementation), including steps, expected results, execution, reviewing and adjusting the test cases and data. |
| **Design Test Case 2(Complex)** | 10 to 11 hours | Design test case 2 (logic implementation), including steps, expected results, execution, reviewing and adjusting the test cases and data. |

## Challenges Faced

#### ****Page Load Delays****:

##### Issue:

### The page load state is taking longer than expected, which necessitates setting multiple timeouts to accommodate varying load times.

### As a result, the overall execution time of the tests has increased significantly.

##### Solution:

### Adjust timeout settings for waiting for page loads or specific elements to load using page.setDefaultTimeout() or increasing timeouts for certain assertions.

### To check for any page or network optimizations that could be made to speed up loading times.

#### ****Frequent Changes in Locators****:

##### Issue:

### The locators used for identifying elements on the page are dynamic and change frequently.

### On average, the locators may work correctly for about 5 test executions, but by the 6th execution, the locators are often altered.

### This inconsistency in locators (those based on auto-generated IDs, dynamic classes, or elements that change frequently, can cause tests to fail after a few successful executions) leads to failures in test executions and requires frequent updates to the test scripts.

##### Solution:

### To use more robust locators based on stable attributes like data-\* attributes or use XPath with specific parent-child relationships that are less likely to change.

### Alternatively, can consider adding custom attributes that help to uniquely identify elements.

#### Flaky Tests

##### Issue:

### Automated tests fail intermittently due to timing issues, such as waiting for elements to load, page state, or network fluctuations.

##### Solution:

### Implement proper waiting mechanisms (e.g., page.waitForSelector, locator.waitFor()), and ensure that the application is in the correct state before interacting with elements. Consider using retries or expect().toBeStable() for more stability.