



DEMO QA TEST PLAN

Juan Tous 07/24

[TEST PLAN]

[Overview]

To expand my job opportunities, I have decided to create this demo with AI on how to apply a Functional and Automation general Test Plan and in another sector.

In this case I chose NASA to apply my experience working with AI . I mean that this is not particularly from SpaceX, Blue Origin or another company.

COPE

NASA has many applications in this case I selected a main system of the ISS called **[ISS Communication Systems]**

This test plan only contains 2 main points about the test plan: there are Design test cases examples and Automation tools...

Note*

This document was made with develop AI open tools

[FUNCTIONAL TEST SUITE]

I have to decided to create test automation suite for the ISS Communication Systems

Test Automation Suite: ISS Communication Systems

Test Scope:

- Verify the Communication Systems' ability to enable communication between the ISS and Earth via radio waves
- Validate the use of Ku-band and S-band frequencies for data transmission

Test Environment:

- Simulation of the ISS Communication Systems hardware and software components
- Emulation of Earth-based communication stations
- Test automation framework (e.g., Python, Java, or C#) with necessary libraries and tools

Test Cases:

Functional Testing

1. Radio Wave Transmission

- Test Case: Verify that the Communication Systems can transmit radio waves to Earth

- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
 - Steps:
 - Send a test signal from the ISS Communication Systems
 - Verify that the signal is received at the Earth-based communication station
 - Expected Result: Signal is successfully transmitted and received
- 2. Ku-band Frequency Transmission**
- Test Case: Verify that the Communication Systems can transmit data using Ku-band frequency
 - Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
 - Steps:
 - Configure the Communication Systems to use Ku-band frequency
 - Send a test data packet from the ISS Communication Systems
 - Verify that the data packet is received at the Earth-based communication station
 - Expected Result: Data packet is successfully transmitted and received using Ku-band frequency
- 3. S-band Frequency Transmission**
- Test Case: Verify that the Communication Systems can transmit data using S-band frequency
 - Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
 - Steps:
 - Configure the Communication Systems to use S-band frequency
 - Send a test data packet from the ISS Communication Systems
 - Verify that the data packet is received at the Earth-based communication station
 - Expected Result: Data packet is successfully transmitted and received using S-band frequency
- 4. Data Transmission Rate**
- Test Case: Verify that the Communication Systems can transmit data at the expected rate
 - Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
 - Steps:
 - Configure the Communication Systems to use Ku-band or S-band frequency
 - Send a large data packet from the ISS Communication Systems
 - Measure the data transmission rate
 - Expected Result: Data transmission rate meets the expected requirements

Performance Testing

1. **Signal Strength**

- Test Case: Verify that the Communication Systems can maintain a strong signal strength
- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
- Steps:
 - Measure the signal strength at the Earth-based communication station
 - Verify that the signal strength meets the expected requirements
- Expected Result: Signal strength is within the expected range

2. **Data Packet Loss**

- Test Case: Verify that the Communication Systems can minimize data packet loss
- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
- Steps:
 - Send a large number of data packets from the ISS Communication Systems
 - Measure the data packet loss rate
- Expected Result: Data packet loss rate is within the expected range

3. **System Latency**

- Test Case: Verify that the Communication Systems can maintain a low system latency
- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
- Steps:
 - Measure the round-trip time (RTT) for data transmission
 - Verify that the system latency meets the expected requirements
- Expected Result: System latency is within the expected range

Security Testing

1. **Data Encryption**

- Test Case: Verify that the Communication Systems can encrypt data transmission
- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
- Steps:
 - Configure the Communication Systems to use encryption
 - Send a test data packet from the ISS Communication Systems
 - Verify that the data packet is encrypted
- Expected Result: Data packet is successfully encrypted

2. **Authentication**

- Test Case: Verify that the Communication Systems can authenticate incoming signals

- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
- Steps:
 - Send a test signal from the Earth-based communication station
 - Verify that the Communication Systems can authenticate the signal
- Expected Result: Signal is successfully authenticated

Compatibility Testing

1. Frequency Interference

- Test Case: Verify that the Communication Systems can operate in the presence of frequency interference
- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
- Steps:
 - Introduce frequency interference in the Ku-band or

7. TEST AUTOMATION TOOLS

Test automation tool Frontend

Playwright was chosen as the test automation frontend tool for this project due to its ability to automate web browsers in a fast, reliable, and scalable way. As a browser automation framework, Playwright provides a high-level API for automating web browsers, allowing us to write tests that interact with our web application as a user would.

The following reasons contributed to the selection of Playwright as our frontend test automation tool:

- **Fast and Reliable:** Playwright is built on top of the browser's DevTools protocol, which provides a fast and reliable way to automate browsers. This ensures that our tests run quickly and consistently, reducing the likelihood of flaky tests.
- **Multi-Browser Support:** Playwright supports all modern web browsers, including Chromium, Firefox, and WebKit, allowing us to test our application across different browsers and ensure cross-browser compatibility.
- **Easy to Use:** Playwright's API is intuitive and easy to use, making it simple to write tests that interact with our web application. Its syntax is similar to Selenium, making it easy for our team to transition to Playwright.
- **Built-in Waiting Mechanisms:** Playwright provides built-in waiting mechanisms, such as `waitForSelector` and `waitForFunction`, which simplify the process of waiting for elements to appear or for specific conditions to be met.

- **Support for Modern Web Features:** Playwright supports modern web features such as web sockets, web storage, and web workers, allowing us to test our application's use of these features.
- **Extensive Documentation and Community Support:** Playwright has extensive documentation and an active community, ensuring that we can get help and support when needed.

Test automation tool Backend

Playwright was chosen as the test automation **API** tool for this project due to its ability to send **HTTP** requests directly from Node.js, allowing for efficient and flexible testing of our application's **API**. This feature enables us to test our server API, prepare server-side state before visiting the web application, and validate server-side post-conditions after running actions in the browser.

Additionally, **Playwright** provides a built-in request fixture that respects configuration options like `baseUrl` and `extraHTTPHeaders`, making it easy to send requests to our API endpoint. Its support for sending all kinds of HTTP(S) requests over the network, including **GET, POST, PUT, and DELETE**, allows us to thoroughly test our API's functionality.

Furthermore, Playwright's ability to reuse authentication state and create new contexts with that state enables us to log in via API calls and then create a new context with cookies already set, streamlining our testing process.

[FUNCTIONAL TEST AUTOMATION SUITE DESING]

Test Automation Suite: ISS Communication Systems

Test Scope:

- Verify the Communication Systems' ability to enable communication between the ISS and Earth via radio waves
- Validate the use of Ku-band and S-band frequencies for data transmission

Test Environment:

- Simulation of the ISS Communication Systems hardware and software components
- Emulation of Earth-based communication stations
- Node.js with Playwright framework for test automation

Test Cases:

Functional Testing

1. Radio Wave Transmission

- Test Case: Verify that the Communication Systems can transmit radio waves to Earth
- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
- Steps:
 - Launch Playwright browser instance
 - Navigate to the Communication Systems simulation page
 - Click the "Transmit Radio Waves" button
 - Verify that the signal is received at the Earth-based communication station
- Expected Result: Signal is successfully transmitted and received

javascript

```
const playwright = require('playwright');
```

```
(async () => {
```

```
const browser = await playwright.chromium.launch();
```

```
const context = await browser.newContext();
```

```
const page = await context.newPage()
```

```
await page.goto('https://iss-communication-systems-simulation.com');
```

```
await page.click('button#transmit-radio-waves');
```

```
// Verify signal reception at Earth-based communication station
```

```
const signalReceived = await page.evaluate(() => {
```

```
  // Check if signal is received at Earth-based communication station
```

```
  return true;
```

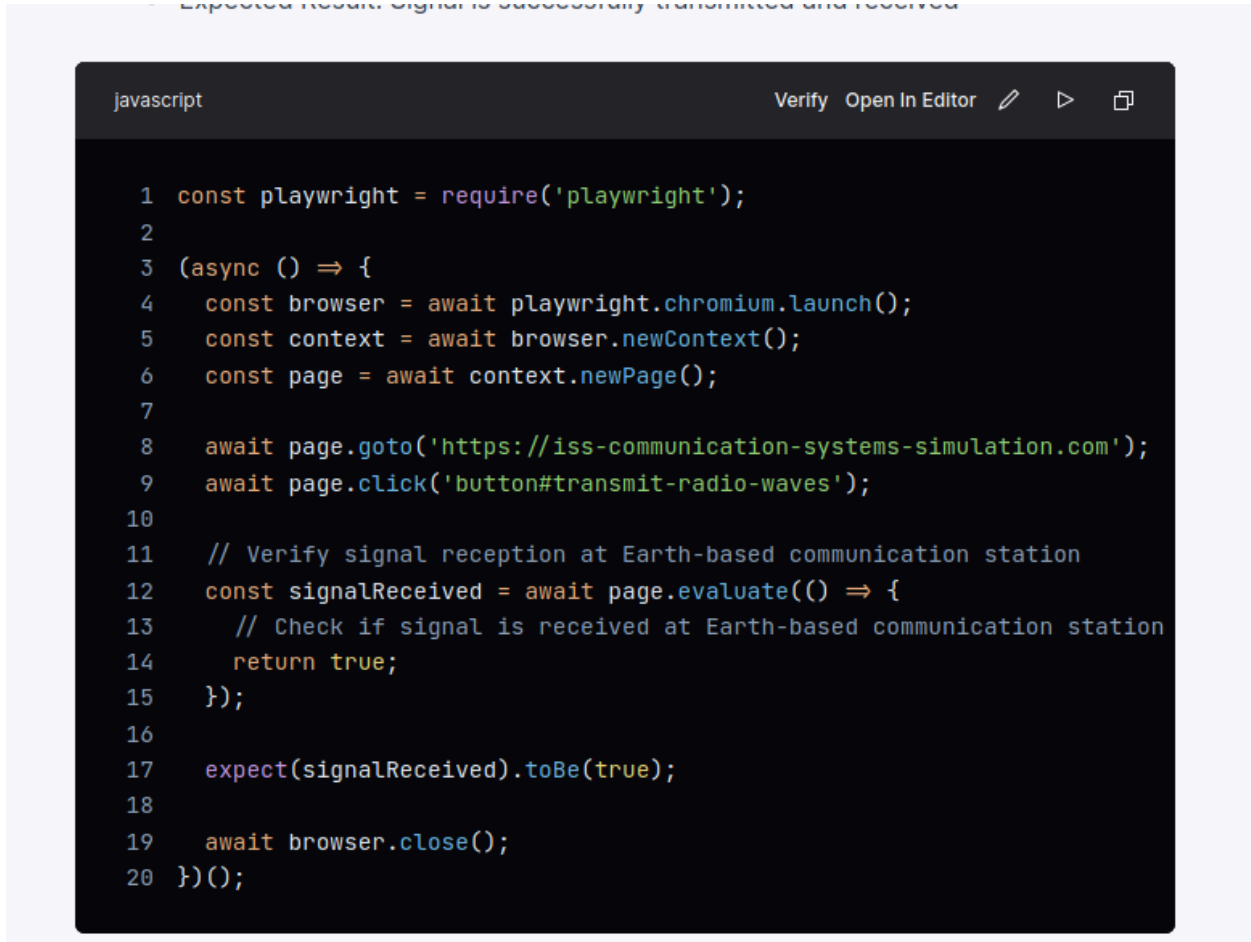
```
});
```

```
expect(signalReceived).toBe(true);
```

```
await browser.close();
```

```
})();
```

Expected Result: Signal is successfully transmitted and received



2. Ku-band Frequency Transmission

- Test Case: Verify that the Communication Systems can transmit data using Ku-band frequency
- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
- Steps:
 - Launch Playwright browser instance
 - Navigate to the Communication Systems simulation page
 - Select Ku-band frequency from the dropdown menu
 - Click the "Transmit Data" button
 - Verify that the data packet is received at the Earth-based communication station
- Expected Result: Data packet is successfully transmitted and received using Ku-band frequency

javascript

```
const playwright = require('playwright');
```


Demo document -non official document
[t.me/Juansitoauth]

```
(async () => {  
  
  const browser = await playwright.chromium.launch();  
  const context = await browser.newContext();  
  const page = await context.newPage();  
  await page.goto('https://iss-communication-systems-simulation.com');  
  await page.selectOption('select#frequency', 'Ku-band');  
  await page.click('button#transmit-data');  
  
  // Verify data packet reception at Earth-based communication station  
  
  const dataPacketReceived = await page.evaluate(() => {  
  
    // Check if data packet is received at Earth-based communication station  
  
    return true;  
  
  });  
  
  expect(dataPacketReceived).toBe(true);  
  await browser.close();  
  
})();
```

```
javascript Verify Open In Editor

1  const playwright = require('playwright');
2
3  (async () => {
4    const browser = await playwright.chromium.launch();
5    const context = await browser.newContext();
6    const page = await context.newPage();
7
8    await page.goto('https://iss-communication-systems-simulation.com');
9    await page.selectOption('select#frequency', 'Ku-band');
10   await page.click('button#transmit-data');
11
12   // Verify data packet reception at Earth-based communication station
13   const dataPacketReceived = await page.evaluate(() => {
14     // Check if data packet is received at Earth-based communication sta
15     return true;
16   });
17
18   expect(dataPacketReceived).toBe(true);
19
20   await browser.close();
21 })();
```

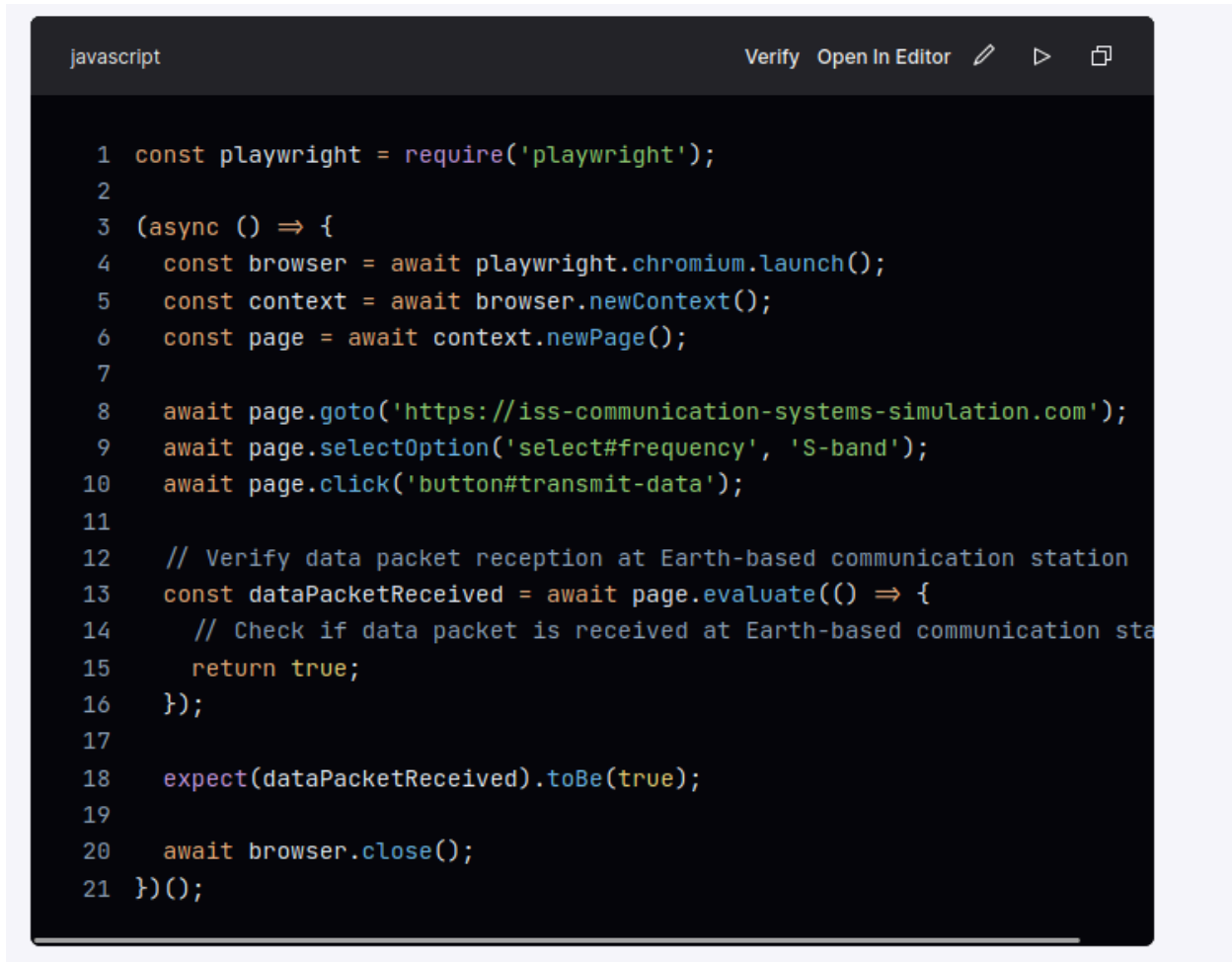
3. S-band Frequency Transmission

- Test Case: Verify that the Communication Systems can transmit data using S-band frequency
- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
- Steps:
 - Launch Playwright browser instance
 - Navigate to the Communication Systems simulation page
 - Select S-band frequency from the dropdown menu
 - Click the "Transmit Data" button
 - Verify that the data packet is received at the Earth-based communication station
- Expected Result: Data packet is successfully transmitted and received using S-band frequency

```
javascript
const playwright = require('playwright');
```

Demo document -non official document
[t.me/Juansitoauth]

```
(async () => {  
  
  const browser = await playwright chromium launch();  
  const context = await browser newContext();  
  const page = await context newPage();  
  await page goto('https://iss-communication-systems-simulation.com');  
  await page selectOption('select#frequency', 'S-band');  
  await page click('button#transmit-data');  
  // Verify data packet reception at Earth-based communication station  
  
  const dataPacketReceived = await page evaluate(() => {  
  
    // Check if data packet is received at Earth-based communication station  
  
    return true;  
  
  });  
  
  expect(dataPacketReceived).toBe(true);  
  await browser close();  
  
})();
```

A screenshot of a code editor window with a dark theme. The title bar shows 'javascript' on the left and 'Verify Open In Editor' with icons on the right. The code is a Playwright test script in JavaScript, numbered 1 to 21. It imports 'playwright', launches a Chromium browser, creates a new context and page, navigates to 'https://iss-communication-systems-simulation.com', selects 'S-band' from a frequency dropdown, and clicks a 'transmit-data' button. It then evaluates a page function to verify data packet reception at an Earth-based communication station and finally closes the browser.

```
1 const playwright = require('playwright');
2
3 (async () => {
4   const browser = await playwright.chromium.launch();
5   const context = await browser.newContext();
6   const page = await context.newPage();
7
8   await page.goto('https://iss-communication-systems-simulation.com');
9   await page.selectOption('select#frequency', 'S-band');
10  await page.click('button#transmit-data');
11
12  // Verify data packet reception at Earth-based communication station
13  const dataPacketReceived = await page.evaluate(() => {
14    // Check if data packet is received at Earth-based communication station
15    return true;
16  });
17
18  expect(dataPacketReceived).toBe(true);
19
20  await browser.close();
21 })();
```

Performance Testing

1. Signal Strength

- Test Case: Verify that the Communication Systems can maintain a strong signal strength
- Preconditions: ISS Communication Systems is powered on, Earth-based communication station is available
- Steps:
 - Launch Playwright browser instance
 - Navigate to the Communication Systems simulation page
 - Measure the signal strength at the Earth-based communication station
 - Verify that the signal strength meets the expected requirements
- Expected Result: Signal strength is within the expected range




```
javascript
const playwright = require('playwright');
```

Demo document -non official document
[t.me/Juansitoauth]

```
(async () => {
```

```
const browser = await playwright.ch
```

javascript

Verify Open In Editor   

```
1 const playwright = require('playwright');  
2  
3 (async () => {  
4   const browser = await playwright.ch
```