Playwright Test Plan for Orbit "Schedule Missions" Page

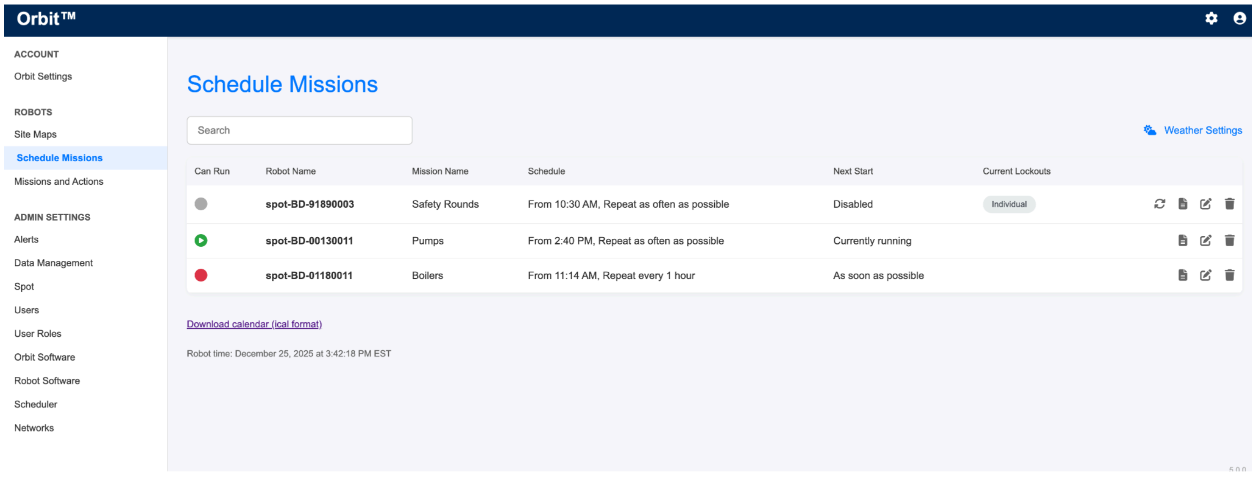
1. Test Objectives

The primary goal of this test plan is to ensure the static Orbit "Schedule Missions" page renders correctly, displays accurate mission data, and supports basic user interactions. Since this is a standalone local HTML file (no backend or dynamic JavaScript), tests focus on:

- UI rendering integrity
- Content validation
- Basic interactivity (e.g., form inputs, links)
- Cross-browser consistency
- Responsiveness
- Accessibility

Secondary objectives include establishing a baseline for future extensions (e.g., when integrating with the Orbit application) and demonstrating reliable, maintainable Playwright test patterns.

Reference Screenshot of the Orbit "Schedule Missions" Page



Orbit™

ACCOUNT
Orbit Settings

ROBOTS
Site Maps
Schedule Missions
Missions and Actions

ADMIN SETTINGS
Alerts
Data Management
Spot
Users
User Roles
Orbit Software
Robot Software
Scheduler
Networks

Schedule Missions

Search

Weather Settings

Can Run	Robot Name	Mission Name	Schedule	Next Start	Current Lockouts
●	**spot-BD-91890003**	Safety Rounds	From 10:30 AM, Repeat as often as possible	Disabled	Individual
▶	**spot-BD-00130011**	Pumps	From 2:40 PM, Repeat as often as possible	Currently running	
●	**spot-BD-01180011**	Boilers	From 11:14 AM, Repeat every 1 hour	As soon as possible	

Download calendar (ical format)

Robot time: December 25, 2025 at 3:42:18 PM EST

5.0.0

Orbit-schedule-missions.html: This is the full-page view of the tested interface, including the header, sidebar navigation with "Schedule Missions" highlighted, mission table with status circles, lockout badge, action icons, search box, Weather Settings link, and footer details such as robot time and version.

2. Scope

In Scope:

- Verification of header, sidebar, main content, table data, and footer elements.
- Validation of mission-specific details (robot names, statuses, lockouts).
- Basic interactions (search input, link clicks, action icons).
- Cross-browser, responsive, visual, and accessibility checks.

Out of Scope:

- Dynamic features (e.g., actual search filtering, API-driven updates, global lockouts), these require a live application.
- Performance testing beyond basic load times.
- Security or authentication flows.

Entry Criteria: Local HTML file (orbit-schedule-missions.html)

Exit Criteria: All critical smoke and functional tests pass with 100% success across targeted browsers/devices; visual baselines established; no accessibility violations.

3. Test Environment Setup

- **HTML File**: orbit-schedule-missions.html (local static file).
- **Playwright Version**: Latest stable (@playwright/test).
- **Browsers**: Chromium, Firefox, WebKit (full cross-browser coverage).
- **Devices**: Desktop (default) + mobile emulation (e.g., iPhone 14, Pixel 7, iPad Air).
- **Execution Modes**: Headed (for debugging), headless (for CI), UI mode (interactive).
- **Dependencies**: Node.js, optional: @axe-core/playwright for accessibility.

All tests load the page via a file URL:

TypeScript
```
const fileUrl = `file://${path.join(__dirname, 'orbit-schedule-missions.html')}`;
```

4. Test Categories & Detailed Test Cases

Tests are organized into categories for maintainability.

A. Legacy / Baseline Tests

These original test scripts provide a simple, foundational verification of page loading and key content. They serve as the starting point for the test suite and can be used for quick local validation in both JavaScript and TypeScript flavors.

JavaScript Version (CommonJS)

This test script demonstrates basic assertions with older locator styles. It includes interaction testing (clicking the Weather Settings link) and screenshot capture.

JavaScript
```
// schedule-missions.test.js
// Playwright end-to-end test for loading a local Orbit Schedule Missions HTML file
// This test verifies that the local HTML file loads correctly and checks for key elements on the page.
// Jaan John

const { test, expect } = require('@playwright/test');
const path = require('path');
```

```
// Resolve the absolute path to your local HTML file
const htmlFilePath = path.join(__dirname, 'orbit-schedule-missions.html');
// Convert to file URL format (file:///...)
const fileUrl = `file://${htmlFilePath}`;

test('Open local Orbit Schedule Missions HTML and verify content', async ({ page }) => {
  // Navigate to the local HTML file
  await page.goto(fileUrl);

  // Wait for the page to load fully
  await page.waitForLoadState('domcontentloaded');

  // Verify the page title
  await expect(page).toHaveTitle('Orbit - Schedule Missions');

  // Verify the main heading
  await expect(page.locator('h1.page-title')).toHaveText('Schedule Missions');

  // Check that the table is visible
  await expect(page.locator('table')).toBeVisible();

  // Verify one of the robot names is present
  await expect(page.locator('text=spot-BD-91890003')).toBeVisible();

  // Verify the "Currently running" status
  await expect(page.locator('text=Currently running')).toBeVisible();

  // Take a screenshot for debugging
  await page.screenshot({ path: 'screenshot-orbit-page.png', fullPage: true });

  // Click on the Weather Settings link (just to test interaction)
  await page.click('text=Weather Settings');

  console.log('All tests passed! Local HTML file loaded successfully.');
});
```

TypeScript Version (ES Modules)

The modern TypeScript equivalent uses import syntax and Playwright's recommended getByText locators for better readability and maintainability.

TypeScript
```
// schedule-missions.test.ts
// Playwright end-to-end test for loading a local Orbit Schedule Missions HTML file
// This test verifies that the local HTML file loads correctly and checks for key elements on the page.
// Jaan John
```

```
import { test, expect } from '@playwright/test';
import path from 'path';

const htmlFilePath = path.join(__dirname, 'orbit-schedule-missions.html');
const fileUrl = `file://${htmlFilePath}`;

test('Open local OrbitSchedule Missions HTML and verify content', async ({ page }) => {
  await page.goto(fileUrl);
  await page.waitForLoadState('domcontentloaded');

  await expect(page).toHaveTitle('Orbit - Schedule Missions');
  await expect(page.locator('h1.page-title')).toHaveText('Schedule Missions');
  await expect(page.locator('table')).toBeVisible();
  await expect(page.getByText('spot-BD-91890003')).toBeVisible();
  await expect(page.getByText('Currently running')).toBeVisible();

  await page.screenshot({ path: 'screenshot-orbit-page.png', fullPage: true });
});
```

These baseline tests are ideal for initial setup verification and can be run directly via commands, npx playwright test schedule-missions.test.js or npx playwright test schedule-missions.test.ts.

B. Smoke Tests (Critical Path)

These confirm the page loads and core elements are present. They run first in any suite to catch fundamental issues early.

The following smoke test suite ensures basic navigation and visibility of key structural elements (title, heading, sidebar active item, table). It serves as a gatekeeper, if smoke tests fail, deeper functional testing is unnecessary.

TypeScript
```
// tests/smoke/schedule-missions-smoke.test.ts
import { test, expect } from '@playwright/test';
import path from 'path';

const fileUrl = `file://${path.join(__dirname, '../../orbit-schedule-missions.html')}`;

test.describe('Smoke Tests - Schedule Missions Page', () => {
  test('Page loads and displays main elements', async ({ page }) => {
    await page.goto(fileUrl);
    await expect(page).toHaveTitle('Orbit - Schedule Missions');
    await expect(page.locator('h1.page-title')).toHaveText('Schedule Missions');
    await expect(page.locator('text=Schedule Missions')).toBeVisible();
    await expect(page.locator('table')).toBeVisible();
    await expect(page.locator('.sidebar li.active')).toHaveText('Schedule Missions');
  });
});
```

C. Functional Tests

These validate content accuracy and basic interactions. The suite below checks the missions table row-by-row, confirms lockout badges, and tests non-navigating elements (since it's static).

This functional suite demonstrates data-driven validation: it asserts exact mission details (e.g., robot names, statuses, lockout badges) and verifies interactive elements behave as expected without causing navigation or errors.

TypeScript

```typescript
// tests/functional/schedule-missions-functional.test.ts
import { test, expect } from '@playwright/test';
import path from 'path';

const fileUrl = `file://${path.join(__dirname, '../../orbit-schedule-missions.html')}`;

test.describe('Functional Tests', () => {
  test('Table displays correct mission data', async ({ page }) => {
    await page.goto(fileUrl);
    const rows = page.locator('tbody tr');
    await expect(rows).toHaveCount(3);

    // Row 1: Safety Rounds (with Individual lockout)
    await expect(rows.nth(0)).toContainText('spot-BD-91890003');
    await expect(rows.nth(0)).toContainText('Safety Rounds');
    await expect(rows.nth(0)).toContainText('Disabled');
    await expect(rows.nth(0).locator('.lockout-badge')).toHaveText('Individual');

    // Row 2: Pumps (Currently running)
    await expect(rows.nth(1)).toContainText('spot-BD-00130011');
    await expect(rows.nth(1)).toContainText('Pumps');
    await expect(rows.nth(1)).toContainText('Currently running');

    // Row 3: Boilers
    await expect(rows.nth(2)).toContainText('spot-BD-01180011');
    await expect(rows.nth(2)).toContainText('Boilers');
    await expect(rows.nth(2)).toContainText('As soon as possible');
  });

  test('Search box is present and interactive', async ({ page }) => {
    await page.goto(fileUrl);
    const search = page.locator('.search-box');
    await expect(search).toBeVisible();
    await search.fill('Pumps');
    await expect(search).toHaveValue('Pumps'); // Input persists (static page)
  });

  test('Weather Settings link is visible and clickable', async ({ page }) => {
    await page.goto(fileUrl);
    const link = page.locator('text=Weather Settings');
    await expect(link).toBeVisible();
    await expect(link).toHaveAttribute('href', '#');
    await link.click();
    await expect(page).toHaveURL(fileUrl); // No navigation
  });

  test('Action icons are present and hoverable', async ({ page }) => {
    await page.goto(fileUrl);
    const firstRowActions = page.locator('tbody tr').nth(0).locator('.actions i');
    await expect(firstRowActions).toHaveCount(4);
    await firstRowActions.nth(2).hover(); // Hover on edit icon (visual check via screenshot)
  });
});
```

D. Cross-Browser Tests

Configure multiple projects in playwright.config.ts for parallel execution across browsers.

E. Responsive / Mobile Tests

Use device emulation to verify layout on mobile viewports.

F. Visual Regression Tests

Capture baselines and compare pixels (reference the inserted screenshot as the expected visual state).

G. Accessibility Tests

Integrate axe-core to scan for violations.

5. Risks and Assumptions

- **Risk** Static nature limits interaction testing; future dynamic integration may require suite expansion.
- **Assumption** HTML file remains unchanged; any UI updates require baseline refresh.

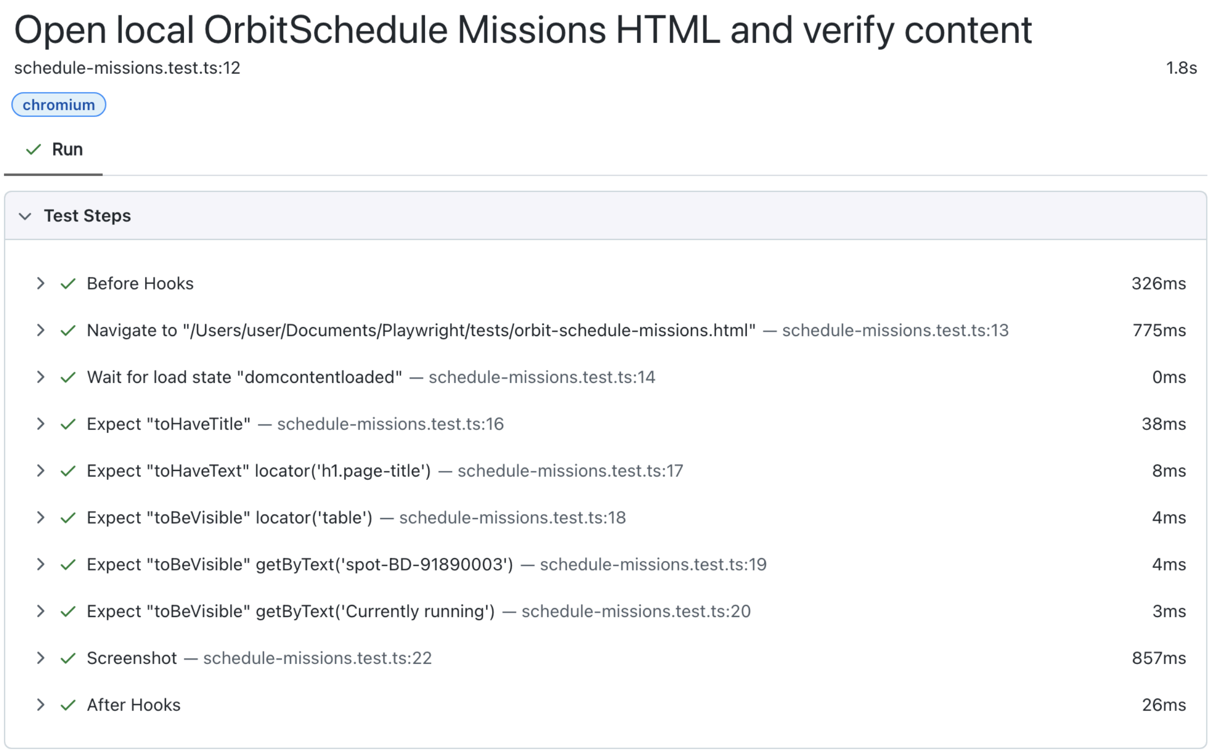
6. Test Execution Strategy

Environment	Command	Purpose
Local Debug	npx playwright test --headed --ui	Interactive debugging
Full Suite	npx playwright test	Run all tests headless
Cross-Browser	npx playwright test --browser=all	Validate across Chromium/Firefox/WebKit

Update Baselines	npx playwright test --update-snapshots	After approved UI changes

7. Reporting

Playwright built-in HTML reporter can be utilized via npx playwright show-report, and below shows the outcome of the test which verifies the pass/fail criteria. Any further screenshots, traces, and videos of failed tests (if applicable) would appear in this section.



Open local OrbitSchedule Missions HTML and verify content

schedule-missions.test.ts:12 1.8s

chromium

✓ Run

Test Steps

```
✓ Before Hooks                                                                                                   326ms
✓ Navigate to "/Users/user/Documents/Playwright/tests/orbit-schedule-missions.html" — schedule-missions.test.ts:13 775ms
✓ Wait for load state "domcontentloaded" — schedule-missions.test.ts:14                                            0ms
✓ Expect "toHaveTitle" — schedule-missions.test.ts:16                                                             38ms
✓ Expect "toHaveText" locator('h1.page-title') — schedule-missions.test.ts:17                                      8ms
✓ Expect "toBeVisible" locator('table') — schedule-missions.test.ts:18                                             4ms
✓ Expect "toBeVisible" getByText('spot-BD-91890003') — schedule-missions.test.ts:19                                4ms
✓ Expect "toBeVisible" getByText('Currently running') — schedule-missions.test.ts:20                               3ms
✓ Screenshot — schedule-missions.test.ts:22                                                                      857ms
✓ After Hooks                                                                                                     26ms
```

8. References:

https://support.bostondynamics.com/s/article/Create-an-Orbit-Mission-Schedule-115575

schedule-missions.test.js
schedule-missions.test.ts
orbit-schedule-missions.html

Jaan John

JaanJohn04@Gmail.com