# QA Automation Assignment – Playwright with Cucumber

This document provides details about the framework choice, automated scenarios, and implementation for the QA Automation Test assignment.

## 1. Choice of Framework & Language

## WebdriverIO is a solid and widely used framework, especially for teams already working with Selenium and needing a broad set of integrations. But for this project, I felt Playwright with JavaScript was a better fit. It talks directly to the browser instead of going through extra drivers, which makes the tests run faster and with fewer flaky failures. Playwright also comes with Chromium, Firefox, and WebKit built in, so setup is really simple. On top of that, features like auto-waiting, screenshots, and trace debugging are available out of the box, which helps a lot during development and debugging. Overall, Playwright gave me a quicker and more reliable way to build the test scenarios for this assignment.

## 2. Automated Scenarios & Implementation

For this assignment, I automated two end-to-end scenarios that represent realistic and critical user journeys across different applications.

### Scenario 1: Create an Employee on OrangeHRM and Verify in Admin Panel

In this scenario, I automated the complete workflow of creating a new employee in the PIM (Personal Information Management) module, saving their details, and then validating the same user in the Admin panel.  
  
I chose this because employee creation and management is one of the most fundamental HR operations. Automating it ensures that not only the PIM module works correctly, but also that data is properly reflected across different sections of the application (cross-module validation).  
  
This flow also covers multiple user interactions like filling forms, toggling account creation, validating generated usernames, and logging out, which makes it a good candidate to test both functionality and UI interactions.

### Scenario 2: User Login and Purchase Flow on BStack Demo Ecommerce

In this scenario, I automated the login of a user, adding an iPhone to the cart, proceeding to checkout, filling in shipping details, completing the order, and verifying the success message.  
  
I selected this flow because purchasing is the core functionality of any ecommerce platform. It validates not just login and cart functionalities, but also order placement and confirmation — which are critical business processes.  
  
This also ensures that the system handles form data correctly (like shipping address), updates the cart state, and completes the purchase lifecycle without errors.

### Implementation & Framework Structure

I followed the BDD approach with Cucumber + Playwright.  
  
The framework is organized as follows:  
- features/: Contains .feature files where each scenario is written in Gherkin.  
- steps/: Maps each Gherkin step to executable Playwright code.  
- pages/: Implements Page Object Model for reusable locators and actions (e.g., LoginPage, DashboardPage, CheckoutPage).  
- support/ or hooks/: Contains setup/teardown logic, utilities, and configuration.  
  
This modular structure makes the framework scalable and easy to extend when adding more test cases in the future.  
Please refer to the Readme.md file in the automation framework to get more information about the setup and dependencies used.