Workshop: Selenium + BDD/Cucumber

Repo: GitHub - DevAnamTales/selenium-with-cucumber: selenium with cucumber workshop

In this workshop we used Selenium with BDD/Cucumber for test automation. Reflecting on the experience, here are my thoughts about this project:

1. **Did I manage to write all the tests (features, scenarios, and step definitions) that I had planned? Why/Why not?**
   Unfortunately, I didn't manage to complete all the planned tests. The main reasons were:
   - **Time constraints:** Creating robust and reusable step definitions required more time than anticipated, especially for dynamic and complex elements like the menu navigation and health bar checks.
   - **Debugging challenges:** Some tests failed unexpectedly due to issues like incorrect locators or timing problems. Fixing these required significant debugging effort.
   - **Learning curve:** Although I have prior experience with Selenium and Cucumber, some cases like win-the-game scenario took longer to implement, but overall I learned a lot during this project.

2. **How does working with Selenium compare to Cypress.io?**
   Advantages of Selenium:
   - **Flexibility:** Selenium supports multiple browsers and programming languages, making it versatile for cross-browser testing.
   - **BDD Integration:** The combination of Selenium with Cucumber offers clear traceability from feature files to implementation, which is great for collaboration.

   Disadvantages of Selenium:

   - **Complexity:** Writing tests in Selenium often feels more verbose compared to Cypress. Locators like XPath or CSS selectors sometimes make tests harder to read and maintain.
   - **Performance:** Selenium tests tend to run slower, especially with animations or heavy DOM interactions.

   Advantages of Cypress.io:

   - **Simplicity:** Cypress has an easier learning curve and a more developer-friendly API. Test scripts are concise and readable.
   - **GUI:** It’s very easy to maintain and debug the tests in cypress especially with the GUI that comes up with cypress. It’s also very easy to run different tests from the cypress GUI

Disadvantages of Cypress.io:

- **Browser limitations:** Cypress primarily supports Chromium-based browsers, limiting its versatility in cross-browser testing.
- **BDD integration:** While Cypress can integrate with BDD frameworks, it's not as seamless as Selenium + Cucumber.

3. **Reflection on working with Selenium and BDD:**
   - I appreciated how BDD helped align tests with user stories, fostering a better understanding of requirements.
   - Selenium's power lies in its ability to handle diverse scenarios, but this comes at the cost of additional complexity and maintenance effort.

DOM Analysis

The structure of the HTML for *Espresso Addict* is well-defined and uses semantic elements, making it suitable for automation testing. Below is a breakdown of the key sections:

1. Header Section

This section displays essential stats for the player.

- **Elements:**
  - **Health:** `<section class="health">` with a progress bar and value (`<span class="val">`).
  - **Money:** `<section class="money">` with a progress bar and value.
  - **Espressos:** `<section class="espressocups">` with a progress bar and value.
  - **Bag:** `<section class="bag">` containing a description of the items inside.

---

2. Main Section

The central area for displaying game visuals and descriptions.

- **Elements:**
  - **Image:** `<img class="big-image">` displays the main image.
  - **Description:** `<p class="description">` contains dynamic game descriptions or prompts.

Challenges while finding locators:

- Some interactive elements like menu choices may require additional steps to ensure they are rendered dynamically before interaction.

- Lack of unique IDs for certain elements means tests rely on class names or XPath, which may be brittle if the DOM changes

---

Conclusion

The DOM structure of *Espresso Addict* provided a good balance between readability and automation potential. Writing step definitions based on this structure was straightforward, especially for value extraction and menu interactions. I learned how helpful it is to write reusable step definitions that simplify automation and improve test maintenance by reducing duplication. Examples include steps for verifying values in progress bars or selecting menu options dynamically based on text.