GitHub Repo

Git clone this repo

https://github.com/stephenmee/mot-playwright-workshop

Exercise 1

Create your starter project

Your Tasks:
1. Create a boilerplate Playwright project
2. Execute the project on your local machine
3. View the report created from your test run
4. Make a failing test, see the options for traces, screenshots, videos
5. Set a baseURL (https://bookcart.azurewebsites.net) for your tests

💡 **Hints:**
- Since we'll be using TypeScript, you'll need nodejs and npm on your system
- Use this command to initialise a new project: npm init playwright@latest
- There are different options for running your test, the simplest is just ***npx playwright test*** (it will run all tests using all projects)
- Options for tracing, screenshots and videos can be found in the playwright.config.ts

Exercise 2

Create a Page Object Model (POM) for the Login Page

Your Tasks:
1. Create page object model for the login page - https://bookcart.azurewebsites.net/login
2. It should include class properties for username input and password input, a class constructor taking 'page' as an argument and a login method to fill in the username and password, finally it should then click the login button.
3. The class should be exported so it can be used in other scripts

💡 **Hints:**
- With baseURL defined in our config, we can just use '/login' to go to the login page
- Refer to Playwright's own online documentation on POMs - https://playwright.dev/docs/pom

Exercise 3

Use the POM in a test fixture

Your Tasks:
1. Create an authenticated user fixture using the POM from the previous exercise
2. Export the fixture so it can be used in other scripts
3. Use the fixture in a new test script e.g. use the logged in page to add an item to the shopping cart

💡 **Hints:**
- Use test.extend() to create custom fixtures
- Fixtures can depend on other fixtures
- Remember to call await use() in fixture implementation
- Remember to export your fixture so it can be re-used elsewhere
- Refer to playwright's online documentation: https://playwright.dev/docs/test-fixtures

Exercise 4

Automate the API requests

Your Tasks:

- Write a test that uses POST endpoint /api/login to check successful login using valid credentials.
- Write a test that validates failure with invalid credentials

💡 **Hints:**

- Check the BookCart API documentation
    - The API endpoints for BookCart are documented and available at the same base URL as the UI:
        - https://bookcart.azurewebsites.net/
    - Use the /api prefix for API endpoints, e.g./api/login
    - Send POST requests
    - Use request.post('/api/login', { data: { ... } }) to validate login.

- Either create a new user by registering in the website or re-use the below

username : motcork
password : Cork123$$

Exercise 5

Add a setup script to store the authenticated browser state, make it a dependency of your project.

Your Tasks:
Create a setup file that stores the logged in state of the application – e.g. it can be stored in *playwright/.auth*

💡 **Hints:**
Refer to https://playwright.dev/docs/auth