Section 1

Outline how you decided to go about this:

Using typescript it allows me to use build-in libraries and functions to do the test. This works when executing via the command line. Follow the step below to execute:

- Prerequisites:
  - Install nodeJs : https://nodejs.org/en/download/package-manager
  - Install typescript globally, this will allow typescript to compile

```
C:\Lee\git\certificateChecker>npm install typescript -g

changed 1 package in 1s

C:\Lee\git\certificateChecker>
```

- Open file directory in command line
- Run the following command, in project directory, to install libraries: npm install
- Run the following command, in project directory, to install dependencies:
  npx playwright install
- Compile typescript file using tsc, this will create a js file.
- Run request using node command

```
C:\Lee\git\certificateChecker>npm install

added 2 packages, and audited 3 packages in 757ms

found 0 vulnerabilities

C:\Lee\git\certificateChecker>tsc CertificateChecker.ts

C:\Lee\git\certificateChecker>node CertificateChecker.js
Attempting CERT Validation : www.ultimateqa.com
Cert Valid From: Jul  7 03:09:51 2024 GMT
Cert Valid to: Oct  5 03:09:50 2024 GMT

C:\Lee\git\certificateChecker>
```

How would you automate this procedure:

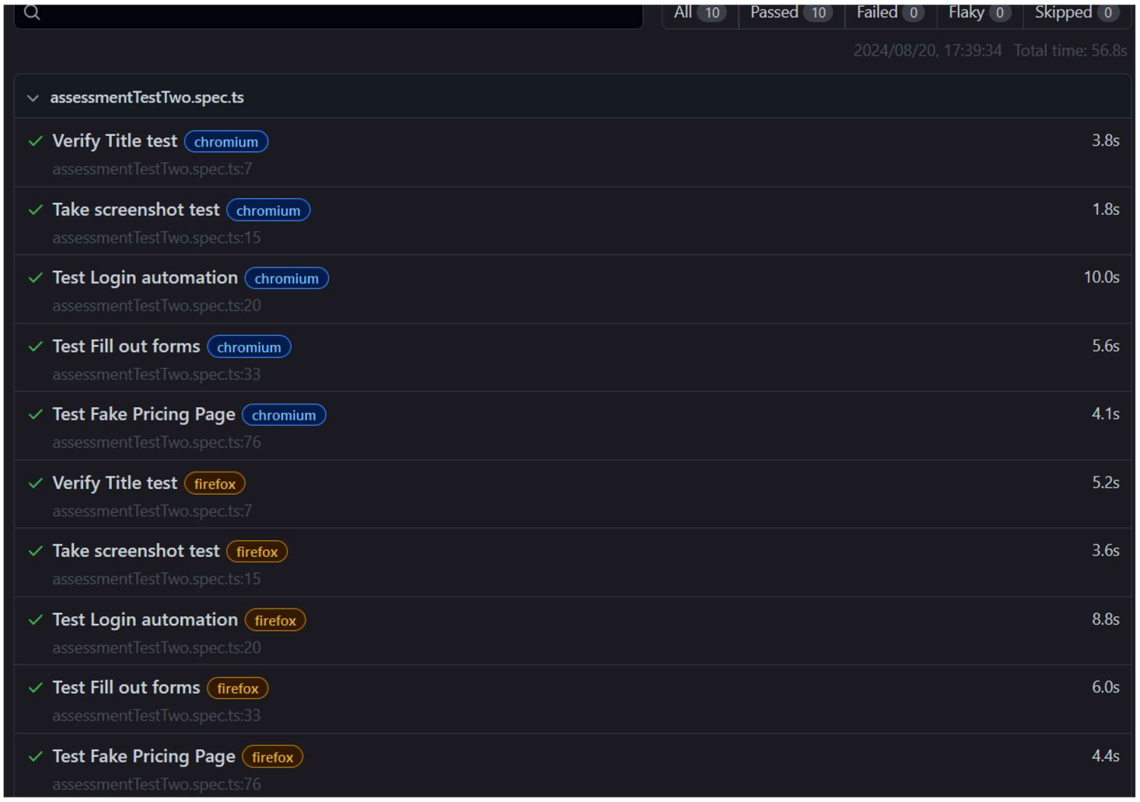
In order to automate this process a cron job can be used. A cron job used for scheduling and automating tasks, at specified intervals, without manual intervention. The javascript file generated above, in section 1, can be used as part of the cron job configuration.

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 2 * * * /root/scripts/encodeIncommingContent.sh >> /tmp/ramdisk/encodeIncommingContent.txt
```

Section 2

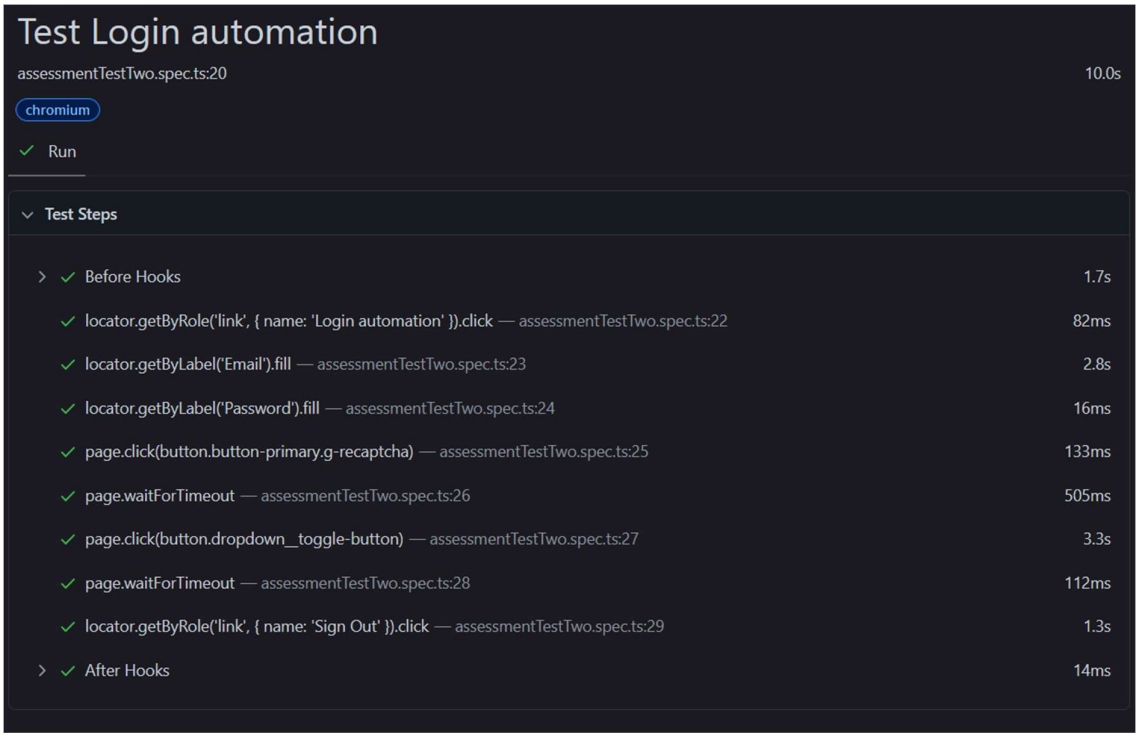
The automation must generate a report:

This can be found in the project directory, playwright-report folder, the file name is index.html.



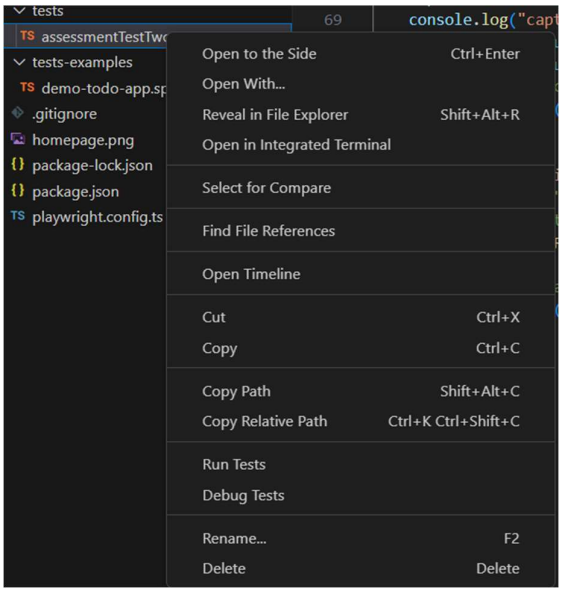
The report can also be viewed from the command line using: npx playwright show-report

An example of login test steps:



Instructions on how to execute the test suite:

a. Project can be executed form with IDE and command line
   - From IDE (this can be seen in the screen recording)
     - Right click on test file
     - Select Run Tests



   - From command line
     - Run “npx playwright test” in the test file directory

Follow up questions

1. In playwright.config.ts the value for fullyParallel can be set to true, this will allow the test to run simultaneously.
   To allow for parallel processing none of the tests should depend on each other.

```
export default defineConfig({
  testDir: './tests',
  /* Run tests in files in parallel */
  fullyParallel: false,
```

```
export default defineConfig({
  testDir: './tests',
  /* Run tests in files in parallel */
  fullyParallel: true,
```

2. In the event of using Jenkins, as CI environment, a job can be configured to kick off a build when a commit is made on the source control system. A step can be added as part of the job to kick off a regressing test pack, that will validate the expected result.
3. I would run multiple instances of the test functionality against an environment.

   Set the number of concurrent workers to more then 1, this can be a value e.g. 10

```
/* Retry on CI only */
retries: process.env.CI ? 2 : 0,
```

   Currently this is set to two concurrent instances of the test.

4. Security testing might include:
   a. Verify unique username, if required
   b. Verify email got valid format
   c. Validate password against requirements
   d. Attempt sql injection into login fields
   e. Make use of captcha

5. Exception handling needs to be in place in order to meet security standards. I will test for the following:
   a. Valid login error message, e.g when username already exists or password invalid
   b. Proper error messages for system related failures, network timeout, database connection failure, etc.
   c. Text entered in a numeric field validation
   d. Check for mandatory fields left blank
   e. In case of calculations, make sure there are proper casting of values