QA Automation Technical Challenge

Skilling is a fintech company with a straightforward purpose: make trading simple and accessible to everyone in a transparent and secure environment.

To avoid performing the same set of manual checks every time we release a new feature or bugfix, we run automated tests against our web page and APIs as part of our CI/CD pipelines. The QA team is responsible for maintaining this suite of test cases to ensure quality across the platform.

In this exercise you should provide tests for the following scenarios:

1. Open the Spanish version of `https://skilling.com` page and check the page title.

Expected values:

- URL is `https://skilling.com/eu/es/`
- Page title is `Skilling — Plataforma de Trading Online`

2. Verify errors returned by the `https://skilling.com/api/v1/sessions` endpoint.

Endpoint info:

- HTTP method: `POST`
- HTTP headers:
  - Accept: application/json
  - Content-Type: application/json
- Fields:
  - username: email
  - password: text

Expected values:

- Response code `422` when `username` or `password` field are not present
- Response code `401` when `username` and/or `password` are valid.

3. Verify price queries using `https://skilling.com/api/public/graphql`

Endpoint info:

- HTTP method: `GET`
- Parameters:
  - query: mandatory graphql expression in the form `{prices(instrumentIds:<array with instruments Ids>) {instrumentId,ask,bid,change}}` . For example, if we need to query for instruments `5156` , `5157` and `5158` we need to perform `GET https://skilling.com/api/public/graphql?query={prices(instrumentIds:[5156, 5157, 5158]){instrumentId,ask,bid,change}}`

Expected values:

- For `GET https://skilling.com/api/public/graphql?query={prices(instrumentIds:[5156, 5157, 5158]) {instrumentId,ask,bid,change}}` a response code `200` with JSON payload like:

```
{
    "data": {
        "prices": [
            {
                "instrumentId": 5156,
```

```
                "ask": number,
                "bid": number,
                "change": number
            },
            {
                "instrumentId": 5157,
                "ask": number,
                "bid": number,
                "change": number
            },
            {
                "instrumentId": 5158,
                "ask": number,
                "bid": number,
                "change": number
            }
        ]
    }
}
```

- For `GET https://skilling.com/api/public/graphql?query={prices(instrumentIds:[-56, 5157, 5158]){instrumentId,ask,bid,change}}` a response code `200` with JSON payload like:

```
{
    "data": {
        "prices": [
            {
                "instrumentId": 5157,
                "ask": number,
                "bid": number,
                "change": number
            },
            {
                "instrumentId": 5158,
                "ask": number,
                "bid": number,
                "change": number
            }
        ]
    }
}
```

Notes

1. A `zip` file has been provided, please extract and work in there. Then zip your solution and send it back.
2. The `zip` contains a folder named `tech-challenge-qa-automation` which has `README.md` and `run-test.sh` files.
3. Keep in mind that code quality is going to be taken in consideration (see clean code, YAGNI and KISS for example)
4. Provide in `README.md` your notes and comments, and any clarifications you wish to make about your approaches or decisions
5. You can use your preferred tools. `Selenium`, `Cypress`, `Spring boot test`, `JUnit`, `Mocha`, `Karma`, `Ruby`, `pytest`, `Cucumber` or any other tool or programming language. Please briefly explain the reasoning behind your choice
6. Your solution must run in a *nix OS. Please modify `run-tests.sh` script to add the commands needed to run your tests. If you are using `Windows` please provide a docker command to execute the tests within a container.
7. Although code for your solution must be limited to what's asked please specify in `README.md` what other tests cases and checks you would consider for the endpoints and language navigation. Don't worry about specifics details like `code error number`, just a high level description other aspects you might check or consider.

8. Please include information about any failing tests in `README.md`, including possible reasons and suggestions on how to fix it.