**WPAY QA CHALLENGE**

**Summary**

We invite you to take this challenge and use the opportunity to express your test automation skills. The challenge contains 2 different exercises related to UI and API test automation.

It is estimated to take around 4-5 hours, but you will have up to 1 week to finish it.

**Requirements**

* Any programming language can be used, recommended Java.
* Any framework/tool can be used, recommended Selenium.
* The use of design patterns is a plus. If implement any, please explain them.
* The solution must be platform agnostic.
* The solution must include test execution report.
* Add a README.md file at the project root with instruction to build the project and run the tests.
* All the software required to build and run the project should be included in the project.
* Once finished send us a mail and attach the solution as a single archive.

**Exercise 1: UI Test Automation**

**Summary**

The goal of this exercise is to cover the UI functionality of login page with appropriate automated tests.

**Description**

Visit Facebook page and validate the login view functionality: https://www.facebook.com.

**The challenge**

1. Define the three most valuable test cases from your point of view. Please explain your choice.
2. Automate the two most important test scenarios. Please justify your choice.

**Exercise 2: API Automation**

**Summary**

The exercise consists of writing API test automation to validate the withdraw API functionality.

**Description**

We are building service that allows users to withdraw money from a company account to an employee's account with following business rules:

1. We have a list of users(employees) (`/v1/users` endpoint).
2. Employees may have at least one payment methods, maxWithdrawalAmount value and maxWithdravals value.
3. A user can execute a withdrawal request using one of the payment methods. A

request must contain one or more withdrawals following the rules:

* 1. Withdrawals count must not exceed the value of `maxWithdrawals` field.
  2. Sum of all withdrawals must not exceed the value of `maxWithdrawalAmount`.
  3. Every withdrawal in the request can be executed immediately (set `executeAt` as now) or scheduled to be executed in future within the current month (set `executeAt` to future date). The field value is expected to be in UTC.

**How the withdrawal processing works:**

1. After accepting a request, the service creates and stores items with status `PENDING` in DB.
2. Then there is a worker that runs every n time (5 seconds by default) and sends all `PENDING` withdrawals before current timestamp to processing.
3. For this test "happy path" is implemented only, so every withdrawal will be updated with status `SUCESSS`.

**The challenge:**

1. Use the `docker-compose.yml` provided in `API Automation` folder.
2. Run the app using `docker-compose up`.
3. The service will be running on port `7070`. There is swagger available by the address: `http://localhost:7070/swagger-ui.html`.
4. Detect all visible problems.
5. Automate three critical scenarios for the immediate withdrawal processing.
6. Suggest a way to create a test automation for the scheduled withdrawal.