How to run tests

# Ho to run UI test

To run the UI tests these tools should be installed along with the application under test:

1. Python 3.7
2. Robot Framework 4.0 (main test UI framework)
3. robotframework-seleniumlibrary (for operating web UI elements)
4. names (python library to generate random names)
5. chromedriver.exe or any other driver to use (for controlling Chrome browser)
6. add chromedriver.exe path in environment variables

To run the tests, you need to run robot command from CMD (***robot UITests/TestSuite/RegistrationTest.robot***)

# Ho to run API test

To run the API tests these tools should be installed along with the application under test:

1. Python 3.7
2. requests (python library for HTTP requests)
3. pytest (python test framework)
4. vcr (python test framework to unit test the API tests)

To run the tests, you need to run pytest command from CMD (***pytest APITests/test\_api.py***).

The tests will add a predefined user needed for the test and then run all API tests (some will fail due to a bug).

To run unit tests for API tests you need to run from CMD (***pytest APITests/test\_api\_with\_vcr.py***)

# Tests Implementation approach

For UI tests I selected Page Object approach as web UI has multiple pages and it will be easy to add new tests or maintain the others. All locators are written as Robot variables in Page files in “UITests /Pages” folder. All actions connected to the page are also there. I tried to use “id” identifier for locators when possible, as “id” is much more stable than the others. All others are CSS or XPATHs. All tests are written in one Test Suite file “RegistrationTest.robot” in “UITests/TestSuite”. Other tests can be added in the same folder. Each time the test is run it will create a random test data using a custom Robot library “NameGenerator.py”, located in “UITests/Utilities” and provide the data for user registration and checking after.

For API tests I used pytest as a main framework. All tests are written in “APITests/test\_api.py”. Before each run the test will create a user in database with one test case from UI tests. Yes, it can sound not loosely coupled but I just showed how I can parametrize a Robot UI test. Ideally it can be done with API test if API has a functionality to POST a user. For unit tests I used vcr tool in python which can be integrated in pytest tests, first run of that tests will create expected scenarios (cassettes) which can be uploaded to Version Control System for future uses. The second run will use mocked scenarios.