ADVANCED SOFTWARE ENGINEERING – LAB TEST – 5th February 2023

NAME: _____ SURNAME: _____ ID (MATRICOLA): _____
ORAL EXAM TOMORROW (IF YOU PASS)? Y: ☐ N: ☐
ORAL EXAM FOR CONTINUOUS ASSESSMENT? Y: ☐ N: ☐
YOUR CODE (Needed in some exercises): 123456

Instructions

It is mandatory to put the number (YOUR CODE) of the header in the first line of the *solutions.txt* file. A missing or misspelt code is considered an automatic fail.

For example, if your code is 123456 the first line of the *solutions.txt* must be:

Your code =:= 123456

You will have 60 minutes to deliver your solution. The deadline is strict, in the course Moodle you will not be able to upload anything after the deadline. If you deliver the solution, you must also give back this sheet.

You can access the slides of the course from the Moodle and the documentation website of PIP (`https://pip.pypa.io/en/stable/`), Docker (`https://docs.docker.com/`), Pytest (`https://docs.pytest.org/en/7.1.x/contents.html`) and Locust (`https://docs.locust.io/en/stable/`).

Any other material, website or application (generative AI included, e.g. Copilot) consulted will result in an automatic fail of the test.

You have to download the zip file of the test and upload an archive file with the requested folders and files as a solution before the deadline.

The test assumes a clean Docker environment containing only the image `python:3.9.18-slim`.

Material description

Download *material.zip* file from the Moodle.

It contains the following:

- folder *Ex1*: a folder containing the code of an application;
- folder *Ex4*: a folder having a `locustfile`, a `docker compose` configuration file, and the code of an application;
- file *solutions.txt*: text file you have to fill with answers, add only text after the =:= symbols without adding new lines, escape symbols, or quotes.

Delivery - 6 points needed to pass

Put all the files and folders (also the unmodified ones) in a `.zip` or `.tar` file and upload it on the Moodle delivery. You can avoid uploading the cache folders created during the execution.

The solutions will be automatically evaluated by an offline script, so be careful to not modify the structure of directories and of the solution file to fill. Add only the answers to the exercises and do not insert new lines, comments, or anything else. To pass this test you need to reach 6 points.
Every exercise but the `Dockerfile` ones, will grant points for partial solutions.
Your delivery must have the following structure:

```
.
├── Ex1
│   ├── app
│   └── Dockefile #from exercise 1
├── Ex4
│   ├── code
│   ├── docker-compose.yml
│   └── locustfile.py
├── solutions.txt
└── Dockerfile #from exercise 2
```

1 Dockerfile 1 (2 points)

In the *Ex1* folder, create and write the `Dockerfile` to make the following image:

- based on the image `python:3.9.18-slim`,
- put (inside the image) the content of the *app* folder inside a folder called *YOUR_CODE* where YOUR_CODE is the number in this sheet header,
- make that folder the working directory,
- use pip to install the requirements of the *requirements.txt* file.

Do not add unnecessary commands, e.g. `apt-get update`. Do not create a directory with `mkdir`. Do not use the flag `--no-cache-dir` when using pip.

2 Dockerfile 2 (2 points)

Build the image of the previous exercise and tag it as `asesecond`. Then, in the exam root folder, create and write the `Dockerfile` to make the following image:

- based on the image `asesecond:latest`,
- use pip to install `pytest` (without specifying the version),
- use pip to install `pytest-cov`,
- make the containers based on the image listen on the port 8000,
- make as the starting command of the container the command to run `pytest` on the `test` folder in verbose mode and with the code coverage test.

As before, do not add unnecessary commands, e.g. `apt-get update`. Do not create a directory with `mkdir`. Do not use the flag `--no-cache-dir` when using pip.

3 Inspect and Pytest (4 points)

Build and run a container based on the image of the previous exercise to generate the `pytest` output.
In the *solutions.txt* file, write the answer to the following:

- write the value of the field `["Config"]["Volumes"]` from the JSON output of the docker command to inspect the executed container,
- write the number of unit tests passed by running `pytest` on the *test* folder,
- write the name of the first test failed (the name is the string between "::" and "-");
- write the tests' coverage percentage of *app.py*.

4 Locust (2 points)

In the *Ex4* folder, run the application with `docker compose`. In a different terminal, run `locust` and use a web browser to reach locust's web service to run performance tests on the application.

Select 100 users, 10 as spawn rate and the base URL of the composed application as host. You can find the port to reach in the *.yml* file.

Analyse the statistics and fill the file *solutions.txt* answering the questions about the endpoint's name with the most requests and the endpoint's name with the most failed requests (wait at least 10 seconds to collect significant statistics).

5 Extra (1 point)

Perform an HTTP GET to the composed application's endpoint `/secret?X=code`, where `code` is YOUR CODE of this sheet header. Look for the HTTP **header** named `exam` in the HTTP response and put its value in the *solutions.txt* file (without any quotes).