Homework 1 – Advanced Software Engineering (2019/20)

Deadline: Oct. 18th, 2019 at 10:59 AM.

Solution code **MUST** be submitted to Moodle as a single `surname_name_hw1.zip` file.

All submissions will be checked for plagiarism. Plagiarised solutions will be awarded an F(0) grade.

Exercise – Fred and Barney need your help to implement the new *Bedrock-a-Millionaire* RESTful service, exposing the API reported on the next page (→). They have already coded a simple `MillionaireSkeleton` based on the microservice skeleton that we have seen in class.

Particularly, Fred and Barney provide you with:

- a `myservice/classes/quiz.py` module, which implements the *Bedrock-a-Millionaire* basic functionalities as plain Python code,
- a `myservice/views/quizzes.py` blueprint, which you must complete to offer all required functionalities of *Bedrock-a-Millionaire* as a RESTful service,
- a `myservice/tests/tests.py` file, which you can run against your solution code by issuing the command `pytest` in the project folder (after running `pip install pytest`).

Download the `MillionaireSkeleton.zip` available from the Moodle and prototype *Bedrock-a-Millionaire*, relying on the *Flask* micro-framework and working on the `myservice/views/quizzes.py` file **only**[1].

The solution must pass all provided tests and must be uploaded to your GitHub.

Write a short report (300 words at most[2]) containing:

(1) the link to the GitHub repository of the project, and
(2) the screenshot of the successful execution of `myservice/tests/tests.py`,
(3) the screenshots of the tests provided in `tests.py`, performed with PostMan for **all** operations.

Upload to the Moodle **both** the report and your new `quizzes.py` file.

Learning Outcomes

- ✓ Revise programming concepts with Python.
- ✓ Revise command-line usage.
- ✓ Get familiar with the Flask microframework and Postman.
- ✓ Get familiar with GitHub.

[1] The API must not be changed nor adapted.

[2] Submitted solutions which exceed the words limit for the report will incur in grading penalties.

URI	ReqType	Description
/quizzes	POST	*Creates a new quiz and gets the quiz identifier back.*
	GET	*Retrieves all active quizzes.*
/quizzes/loaded	GET	*Returns the number of quizzes currently loaded in the system*
/quiz/<id>	GET	*Retrieves the quiz identified by* <id>.
	DELETE	*Deletes the quiz identified by* <id> *from the system.*
/quiz/<id>/question	GET	*Retrieves the current question from the quiz identified by* <id>.
/quiz/<id>/question/<answer>	PUT	*Checks if the given* <answer> *is correct for the current question of the quiz identified by* <id>.