Homework 1 – Advanced Software Service Engineering (1st Term 2018)

Deadline: Oct. 15th, 2018 at 1:59 PM.

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-Doodle* RESTful service, exposing the API reported on the next page (→). They have already coded a simple `DoodleSkeleton` based on the microservice skeleton that we have seen in class.

Particularly, Fred and Barney provide you with:

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

Download the `DoodleSkeleton.zip` available from the Moodle and prototype *Bedrock-a-Doodle*, relying on the *Flask* micro-framework and changing the `myservice/views/doodles.py` file **only**.

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

Write a short report (300 words at most[1]) containing:

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

Upload to the Moodle **both** the report and your solution code.

Learning Outcomes

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

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

Homework 1 – Advanced Software Service Engineering (1st Term 2018)

URI	ReqType	Description	Example Input JSON	Example Output JSON
`/doodles`	`POST`	*Creates a new poll and gets the poll identifier back.*	`{ "title": "pool", "options": ["mon", "tue", "wed"] }`	`{ "pollnumber": 3 }`
	`GET`	*Retrieves all active doodles as a list.*		`{ "activepolls": [ { "id": 1, "options": { "mon": [], "tue": [], "wed": [ "fred" ] }, "title": "pool", "winners": [ "wed" ] }, { "id": 2, "options": { "mon": [], "tue": [], "wed": [] }, "title": "pool", "winners": [ "mon", "tue", "wed" ] } ] }`
`/doodles/<id>`	`GET`	*Retrieves the doodle identified by* `<id>`.		`{ "id": 1, "options": { "mon": [], "tue": [], "wed": [ "fred" ] }, "title": "pool", "winners": [ "wed" ] }`
	`PUT`	*Votes and returns the list of currently winning options.*	`{ "person": "fred", "option": "wed" }`	`{ "winners": [ "tue", "wed" ] }`
	`DELETE`	*Deletes a Doodle from the system. Returns the list of winning options.*		`{ "winners": [ "tue", "wed" ] }`
`doodles/<id>/<person>`	`GET`	*Retrieves all preferences expressed by* `<person>` *in poll* `<id>`.		`{ "votedoptions": [ "tue", "wed" ] }`
	`DELETE`	*Deletes all preferences expressed by* `<person>` *in poll* `<id>`*. Returns False, when no vote is found for* `<person>`.		`{ "removed": true }`