**Enterprise System Integration UT 2021 -  
Homework 1**

# **General Notes**

This homework will be done in your team (3-4 persons). You are encouraged to distribute tasks among yourselves, but of course, challenging tasks can also be carried out by multiple persons at the same time or might even demand for it. As mentioned in the first class, we expect all the tasks here in the homework to be reviewed or audited by another member of the class (preferable another team member).

In this review/audit, please comment on the following questions (the main idea here is to give real feedback to the implementers - praise and constructive criticism):

* Name(s) of person(s) who implemented the solution for the task
* Name of reviewer
* Was the stated problem/task solved in an acceptable manner? (Would this count as a delivery for a potential client?) - if this is a no, the task is rejected and can be redone (this option can be taken four times in total - has to be discussed with instructors).
* What has been done well and why?
* What is not well implemented and why?
* How easy is it to understand (what is easy to understand/well documented - what is hard to understand/lacks documentation)?
* Are there any recommendations to simplify some of this task that the reviewer would like to share?
* Did the implementers use anything special that was notable for the reviewer and should eventually be shared with a bigger audience? Note these specialties down with a positive remark to encourage sharing.
* Anything else notable, encouraging, or funny?

The review needs to be accompanying the homework submission (best embedded in the text or linked from a content document) and should also go into the log of the reviewer to apply for extra points for the reviewing work at the end of the class. Developed code needs to be linked from the submission document and should be in a git repository. (Only) One of your team members needs to submit the overview document in Moodle before the deadline.

# **Task 1**

Create a simple backend service in Golang that exposes a REST interface for a TODO app with the following functionality:

* List ToDos
* Add ToDo
* Remove ToDo
* Mark ToDo as completed
* Add a crazy, artsy, or funny feature that makes your TODO app unique (e.g. colorization or importance levels, creation date, emoji counters, ... anything else not too complex is welcome here) - please document what you picked!

# **Task 2**

Create a simple front-end app (build a Golang console based front-end package) for the ToDo app (see Task 1) with the following features:

* List ToDos
* Add ToDo
* RemoveToDo
* Mark ToDo as completed
* Support for the special feature from Task 1 (you might have to talk to or even collaborate a bit with the person designing task 1)

# **Task 3**

1. Implement functional\* tests for the backend service.
2. Dockerize the back-end service and the front-end app.
3. Create a docker-compose that is able to run the functional tests for the backend service.
4. Create integration tests on the front-end side that assume a running backend service in the background.
5. Update the docker-compose to accommodate the integration tests.

\* by functional we refer to external/black-box style tests

**Task 4**

1. Setup a Docker registry in your cloud machine.
2. Push the backend image and the front-end app to the Docker registry.
3. Run the docker-compose setup in your cloud machine, making sure the images of the backend service and front-end app are pulled from the Docker registry.
4. Research into techniques to optimize the size of the Docker images and try to reduce the size.

# **Task 5**

Docker is a convenient way to run software. Explore the DockerHub (hub.docker.com) to find an image of preference. You may consult yourself with the awesome-selfhosted list (<https://github.com/awesome-selfhosted/awesome-selfhosted>).

1. Figure out how to run the image that you found.
2. Create a new docker-compose config containing the backend service from Task 1. Add a proxy of your choice to the docker-compose, such that services are served in the following manner:

* Backend service: localhost:8080/api
* YourChosenImage: localhost:8080/mystery

You could use subdomains as well e.g.

* api.localhost:8080
* mystery.localhost:8080