**Analysis phase 2**

**Timeslot: 1530 – 1630**

**Introduction**

In this first analysis phase, you have two tasks to perform. You will continue working on the application <https://github.com/anilallewar/microservices-basics-spring-boot>, the same you used in the discovery phase and analysis phase 1. **IMPORTANT**: **perform the tasks in order**, starting from the top.

You also received files with the output produced by the tool for these tasks. **IMPORTANT:** **only use them when the tasks say so**. This is a fundamental requirement of this evaluation. Apart from that, you can use all your preferred analysis tools, Google, etc.

**Task 3**

**START TIME** – Please fill in the current time: \_\_\_\_\_\_\_\_

(a) First, evaluate the following rule **without using the tool’s outputs**:

Rule: *All connections between a service and an external entity should be encrypted.*

Explanation: An external entity is a system component for which there is no implementation code or deployment configuration found in the analysed application’s source code. For example, mail servers, clients, or databases.

Question: please analyse the code to check if the application fulfils this rule or violates it. We are interested in the details of how you reach your conclusion and the places in the code that prove this conclusion. If you can, please provide link(s) (e.g., files, line numbers) to GitHub that point to those places that prove your answer.

YOUR ANSWER HERE:

**MID TIME** – Please fill in the current time: \_\_\_\_\_\_\_\_

(b) After you are done with the manual analysis in part (a), you can **now open the file** *tool\_output\_rule3.html*. It contains the result of executing the following query on the model with the tool:

Query: *edges.sender\_is("external\_entity").receiver\_is("service").all\_are("encrypted")  
.AND(edges.sender\_is("service").receiver\_is("external\_entity").all\_are("encrypted"))*

The query is the rule from part (a) expressed in the tool’s rule language. It selects all edges of which the sending node has the annotation *external\_entity.* From the resulting edges it selects those, where the receiver has the annotation *service*. For the remaining edges, it checks whether they all have the annotation *encrypted*. Finally, it does the same for connections in the other direction and combines the results with a logical operator *AND*.

Question: Can you verify that the tool’s output is correct? Did you reach the same verdict? Would you give the same evidence as the tool does?

YOUR ANSWER HERE:

**END TIME** – Please fill in the current time: \_\_\_\_\_\_\_\_

**Task 4**

**START TIME** – Please fill in the current time: \_\_\_\_\_\_\_\_

You can **now open the file** *tool\_output\_rule4.html*. It contains the result of executing the following query on the model with the tool:

Query: *nodes.that\_are("entrypoint").all\_have("load\_balancer")*

The query corresponds to the rule:

Rule: *All entry points should have a load balancer.*

The query selects all nodes that have the annotation *entrypoint* and checks, whether they all have the annotation *load\_balancer*.

Question: Can you verify that the tool’s output is correct? Do you reach the same verdict? Would you give the same evidence as the tool does?

YOUR ANSWER HERE:

**END TIME** – Please fill in the current time: \_\_\_\_\_\_\_\_