LO4 Supporting document

4.1 Identifying gaps and omission in the testing process.

Because of the lack of time and resources, there have been gaps and omissions in the testing process, despite it being a good start. For unit and integration tests, the main gap is that we **didn’t test with different types of data**. The testing we did was done using only tests with a single set of data. It would be beneficial to **test the system with a variety of different types of data, such as large numbers of orders or orders with different statuses** to ensure the system can handle different scenarios. Furthermore, we only manually created a few users and orders. It would be great, with more resources, to maybe use a model-based testing approach, or to **automatically generate loads of synthetic data, with a wide range of inputs**. For example, we didn’t try providing a very long description to the bounderies limits.

We also **didn’t test with different environments**: our testing process does not include testing the system in different environments, such as staging or production to ensure the system can handle different configurations.

Regarding performance testing, **we didn’t model the system’s performance characteristics to identify potential bottlenecks**. Our testing limited itself to providing general metrics, such as the number of requests, the types of responses etc… Furthermore, we didn’t have enough resources and the right tools to get precise metrics, such as obtaining the memory/CPU usage. Additional libraries could be used in order to get these metrics. Our testing doesn’t also test for sudden spike in traffic or a DDoS attack that could reveal weaknesses in the system’s ability to handle unexpected loads. Finally, our testing doesn’t provide detailed performance reports that could be helpful for system administrators.

4.2 Identifying target coverage/performance levels for the different testing procedures

For unit and integration testing, the **target coverage would be 100%, including all possible input and scenarios**. For the target performance, all functions should execute with no errors or unexpected behaviour. The tests should also be able to execute in a short amount of time to allow devs to run these tests multiple times without time being a problem. Regarding performance testing, the target coverage would be to identify all potential bottlenecks or scalability issues that could impact the system's performance. It should identify any issues or inefficiencies in the system's design or implementation that could impact the system's performance. It should also handle different types of load, including number of concurrent users and requests per second, and should handle the defined load with response times and error rates within acceptable limits. Finally, it should test the system's ability to provide detailed performance reports.

4.3 Discussing how the testing carried out compares with the target levels

During unit and integration testing, for some files, we achieved 100% code coverage, such as for the files in the models. However, we didn't manage to get coverage metrics for the files in the endpoints folder. With appropriate resources, we would use a specific library in order to get these metrics. In addition, we met the target that tests should execute in a short amount of time. All the tests passed with great coverage, meaning that the requirement has been tested thoroughly. Regarding performance testing, we limited ourselves to providing basic metrics, but these metrics aren't enough to identify bottlenecks in our system. Indeed, a lot of additional work would need to be done in order to isolate the various parts of the systems, modify them, and re-run performance tests.

4.4 Discussion of what would be necessary to achieve the target levels

Despite having met some of the targets, we didn't meet the target that we should use a wide variety of inputs for unit. As explained earlier, the users and orders we added to the database were designed by hand, thus won't cover as much scenarios as if they were automatically generated. Regarding performance testing, in order to achieve or exceed the target levels, we would need additional tools for monitoring metrics, in order to understand precisely where the system slows down under heavy load. We would then need more resources, such as developers, in order to adapt the system based on the results, and re-iterate to analyse the changes and assess any improvement.