Test Report

Test Plan VALIDATION

Nedelchev,Tsanko T.N.

2023

Contents

[Introduction 2](#_Toc138019464)

[Unit Testing 2](#_Toc138019465)

[Integration testing 3](#_Toc138019466)

[Quality Testing 3](#_Toc138019467)

[Performance Testing 4](#_Toc138019468)

[Security testing 6](#_Toc138019469)

[Conclusion 8](#_Toc138019470)

# Introduction

The purpose of this document is to provide an answer to the test plan that has been defined for this project. This report goes through all the tests defined in the test plan and shows the results of the tests that have been performed based on that plan and the acceptance criteria This report defines whether the test was a success or a failure.

# Unit Testing

A screenshot of a computer

Description automatically generated

Unit tests have been defined for the service layer of the auth microservice and the postfeed microservice. The tests run and all pass.

Therefore, this test is a success, based on the criteria defined in the Test Plan.

# Integration testing

Integration testing via Cypress as described in the test plan was not performed due to lack of time and therefore this test is considered a failure.

# Quality Testing

Quality Testing as defined in the test plan should be covered by SonarQube. SonarQube is setup in the pipeline and it triggers a scan on every push to the dev branch.

A screenshot of a computer

Description automatically generated with medium confidence

The pipeline runs through successfully and triggers a SonarQube Scan.

A screenshot of a computer

Description automatically generated with low confidence

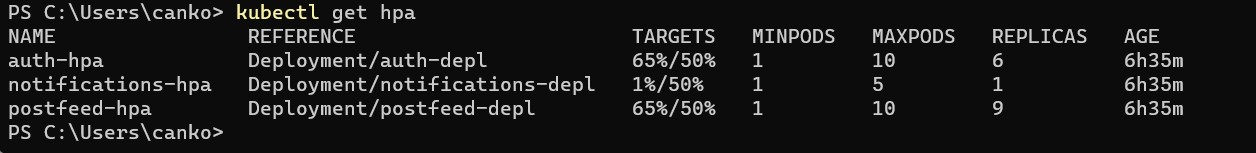
The scan fails due to the Unit Test coverage as the default value is 80% however based on the acceptance criteria from the test plan the test is still a success as there are no bugs present.

# Performance Testing

A test was performed using Apache JMeter on the deployed cluster. The test consisted of 1000 users making requests every second to the two back end APIs.  
  
A screenshot of a computer

Description automatically generated

The scan is triggered and the test begins. This is equivalent to 2000 requests every second.



The results of the scan indicate that the pods for the auth api and the postfeed api got under heavy load and started scaling horizontally.

A picture containing line, plot, diagram, parallel

Description automatically generated

The CPU usage on the VM went way up and triggered an alert. The alert was sent to my email as a notification.

|  |  |
| --- | --- |
| A screenshot of a computer error  Description automatically generated with low confidence | Judging by the criteria set in the test plan this test was a success. |

# Security testing

The security test defined in the test plan indicated that the deployed application should be scanned with OWASP ZAP which triggers through the pipeline.

A picture containing screenshot, text, multimedia software

Description automatically generated

The pipeline setup for deployment and OWASP ZAP scan works in the following way. After the deployment is finished the buffer task begins. The buffer task is a buffer that waits for exactly 10 minutes after the deployment before it triggers the OWASP ZAP scanner. The reason for that distinction is that the cluster needs some time to setup after a deployment. That’s why a healthy window of 10 minutes is required. After the cluster has scaled down and stabilized the OWASP ZAP scan is triggered and scans the application for 8 minutes. After it has finished any new issues are automatically added to the GitHub issue board where they await a checkup by the developers.

A picture containing text, font, screenshot, graphics

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

The OWASP ZAP report after a scan. The test is considered a success despite the high amount of issues most of them are false. One issue that needs to be fixed in the future is the Anti-CSRF token. It would be nice to fix that issue in the future in order to make the cookies of the application more secure.

During another manual scan using OWASP ZAP I discovered a vulnerability that allowed for directory browsing on the client application:  
A screenshot of a computer

Description automatically generated with medium confidence

After some research I fixed the vulnerability using the Ingress-Nginx setup in the Kubernetes manifests:  


This snippet is an NGINX configuration directive that disables directory listing

# Conclusion

After conducting the tests, it can be concluded that all experiments yielded positive results except for the Cypress test. Regrettably, the Cypress test was not performed due to time constraints.