**Test Report**

**Version 1.0**

DevSecOps Team

Spring 2022

University of Maryland Global Campus

SWEN 670

Prepared by Robert Wren and Andrew Nicolette

For Approval by Dr. Mir Mohammed Assadullah

**Revision History**

The Revision History lists specific changes made to this document, dates of change, the person or teams who made the changes, and references to the affected pages/sections/paragraphs.

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author(s)** |
| 04/02/2022 | 1.0 | Initial document | Robert Wren  Andrew Nicolette |

**Table of Contents**

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

2. Testing 5

2.1 Test Strategy 5

2.2 Tests Conducted 5

3. Functional Test Execution Findings 6

3.1 Example Finding 1 6

4. Test Results and Summary 6

4.3.1 Test Case: Version Control - Merge Development Branch (Developer) 7

4.3.2 Test Case: Version Control - Merge Development Branch (Lead Developer) 8

4.3.3 Test Case: Version Control - Merge main branch (Development Team) 9

4.3.4 Test Case: Version Control - Merge Main Branch (DSO) 10

4.3.5 Test Case: Application Build - Manual Execution 11

4.3.6 Test Case: Application Build - Automated Execution 12

4.3.7 Test Case: Application Test - Manual Execution 13

4.3.8 Test Case: Application Test - Automated Execution 14

4.3.9 Test Case: Application Analysis - Manual Execution 15

4.3.10 Test Case: Application Analysis - Automated Execution 16

4.3.11 Test Case: Triggers - Enable Continuous Integration (CI) 17

4.3.12 Test Case: Triggers - Disable Continuous Integration (CI) 18

4.3.13 Test Case: Application Deployment - Manual Execution 19

4.3.14 Test Case: Application Deployment - Automated Execution 20

4.3.15 Test Case: Triggers - Enable Continuous Delivery (CD) 21

4.3.16 Test Case: Triggers - Disable Continuous Delivery (CD) 22

4.4 Example Android Pipeline Test Results 23

4.5 Example iOS Pipeline Test Results 24

# 1. Introduction

## 1.1 Purpose

This test report shall lay out the scope, test strategy, test suite, test execution findings, if any, and metrics for a single test session of the application under test, the MemorEZ DevSecOps Continuous Integration/Continuous Deployment (CI/CD) pipeline within GitHub Actions. This document therefore shall be a summary of the conducted test cycle and include the activities, details, and results of said testing. The test report is a deliverable that can be utilized by stakeholders to evaluate and/or measure the application’s quality. Results are a representation of the state of the application in its current version. Defects identified, if any, can be analyzed to inform for remediation, improvements, or next steps before release. If corrective action is required, the development team can use the information provided in the “Findings” section to aid in resolution.

## 1.2 Scope

This scope of the test cycle shall include core functionality of the Continuous Integration/Continuous Deployment pipeline and its various use case scenarios. Verification of the system functionality includes use of configurations and options set by the DevSecOps team to enable, disable, or control the flow of development operations processes, in order to confirm actual output compared to expected behavior. This shall verify that the CI/CD pipeline meets requirements as specified. Out-of-scope components include, and are not limited to, external testing activities such as usability testing, penetration testing, and performance testing of the proprietary and open source tools used by the project.

# 2. Testing

## 2.1 Test Strategy

A Continuous Integration (CI) pipeline with a centralized code repository will be developed for the two application development teams with test automation and static code analysis tools to verify functionalities and ensure industry standards are followed, including security. A Continuous Delivery (CD) pipeline will be developed to build and deploy the source code to the desired target environment.

Each test case run shall be a single run of the pipeline features, run in the web browser utilized for testing after logging into the DevSecOps tools with the required credentials. This is black-box testing using the supplied user interfaces of the various DevSecOps tools to verify the requirements as set forth in the Test Report document. The test cases focus on use case scenarios by DevSecOps team members as well as development team members as they utilize the DevSecOps infrastructure. The test suite revolves around configurations of access permissions and usage of manual and automated triggers within the context of the integration, build, testing, and deployment functionalities. Testing the functionality from both DevSecOps and development teams perspectives ensures that the core feature of the infrastructure is working as expected. The tester shall ensure the prerequisites are met as outlined in the “Prerequisites” column, and follow the steps outlined in the “Steps” column to execute each specific test case. Test case actual output that matches with expected output shall be determined to be a “Pass”, whereas output that does not match shall be marked as “Fail”, details of which are included in forthcoming sections 2.2 and 3. Test cases that were not able to be run or could not provide conclusive evidence for either pass or fail shall be marked as “N/A".

## 2.2 Tests Conducted

The test suite for the MemorEZ DevSecOps CI/CD pipeline consists of sixteen (16) test cases as laid out in the MemorEZ DevSecOps Test Report document, represented in the tables of test cases present in section 4. Descriptions of each test case are included in the “Description" column and configurations tested are in the “Prerequisites” column. Additional information includes the test case ID and name, requirements tested against, expected results, actual results, and status of the test, along with any notes.

# 3. Functional Test Execution Findings

Given the support nature of the DevSecOps team, it is logical that any proposed internal project schedule put forth is significantly influenced by given project teams' ability to meet and deliver artifacts on schedule. As such, the DevSecOps team reserves the right to make changes to the deliverable timeline within five (5) days of a set milestone, utilizing the above communication plan.

## 3.1 Example Finding 1

Test Case: N/A

Failing Requirement: N/A

Expected Output: N/A

Actual Output: N/A

Defect: N/A

Screenshot of failure: N/A

## 4. Test Results and Summary

The complete test suite was run several times over the last development cycle. The first

run was executed when the CI/CD pipeline was still under development, leading to a

portion of the test cases not being able to run. In the final state at the end of the

milestone, most of the remaining test cases regarding the release pipeline was able to be

run and the results updated for subsequent test runs.

At this time, there are no failures noted in the testing of the infrastructure and pipelines

and test cases conformed to the verification of the requirements as stated in the MemorEZ DevSecOps SRS document.

### 4.3.1 Test Case: Version Control - Merge Development Branch (Developer)

|  |  |
| --- | --- |
| **Description** | Verify merging to development branch by a non-lead developer requires a pull request and code review/approval by lead developer. |
| **Requirements** | Feature/bug branch ready to merge. |
| **Prerequisites** | The development team’s repository is set up with the required branch protection rules and the developer has invited account to the organization and team. |
| **Test Steps** | 1. As developer, create code change and make an integration/feature branch. 2. Push the code to the remote repository. 3. Attempt merge to development branch |
| **Expected Output** | Merging to development branch by non-lead developer is not possible. Developer must create a pull request for review, which notifies the developer lead. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.2 Test Case: Version Control - Merge Development Branch (Lead Developer)

|  |  |
| --- | --- |
| **Description** | Verify merging to development branch by a lead developer can be done without additional pull request or code review. |
| **Requirements** | Feature/bug branch ready to merge. |
| **Prerequisites** | The development team’s repository is set up with the required branch protection rules and the developer lead has invited account to the organization and team and elevated permissions as set by the DSO team. |
| **Test Steps** | 1. As developer lead, create code change and make an integration/feature branch. 2. Push the code to the remote repository. 3. Attempt merge to development branch. |
| **Expected Output** | Merging to development branch by lead developer is successful. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.3 Test Case: Version Control - Merge main branch (Development Team)

|  |  |
| --- | --- |
| **Description** | Verify merging to main branch by the development team requires a pull request and code review/approval by the DSO team. |
| **Requirements** | Feature/bug branch ready to merge. |
| **Prerequisites** | The development team’s repository is set up with the required branch protection rules and the developer has invited account to the organization and team. |
| **Test Steps** | 1. As developer, attempt merge to main branch. |
| **Expected Output** | Merging to main branch directly by development team is not possible. Developer must create a pull request for review, which notifies the DSO team. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.4 Test Case: Version Control - Merge Main Branch (DSO)

|  |  |
| --- | --- |
| **Description** | Verify merging to main branch by DSO team can be done without additional pull request or code review. |
| **Requirements** | Feature/bug branch ready to merge. |
| **Prerequisites** | The development team’s repository is set up with the required branch protection rules and the DSO team member has elevated permissions as set by the DSO team. |
| **Test Steps** | 1. As DSO team member, attempt merge to main branch. |
| **Expected Output** | Merging to main branch by DSO team is successful. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.5 Test Case: Application Build - Manual Execution

|  |  |
| --- | --- |
| **Description** | Verify the CI/CD pipeline application build tasks run when the build feature is manually executed. |
| **Requirements** | Feature/bug branch ready to build. |
| **Prerequisites** | The CI pipeline is integrated with the remote code repository. |
| **Test Steps** | 1. As DSO team member, manually kick off the application build tasks. |
| **Expected Output** | Application build tasks run successfully and notifications are sent. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.6 Test Case: Application Build - Automated Execution

|  |  |
| --- | --- |
| **Description** | Verify the CI/CD pipeline application build tasks run when changes are made to the repository’s main branch. |
| **Requirements** | Feature/bug branch ready to build. |
| **Prerequisites** | The CI pipeline is integrated with the remote code repository. |
| **Test Steps** | 1. As DSO team member, merge integration branch into main branch. |
| **Expected Output** | Application build tasks run successfully and notifications are sent. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.7 Test Case: Application Test - Manual Execution

|  |  |
| --- | --- |
| **Description** | Verify the CI/CD pipeline application test tasks run when the application testing feature is manually executed. |
| **Requirements** | Feature/bug branch ready to build with unit tests complete. |
| **Prerequisites** | Test code is integrated with the CI pipeline. |
| **Test Steps** | 1. As DSO team member, manually kick off the application test tasks. |
| **Expected Output** | Application testing tasks run successfully and notifications are sent. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.8 Test Case: Application Test - Automated Execution

|  |  |
| --- | --- |
| **Description** | Verify the CI/CD pipeline application test tasks run after build tasks are complete. |
| **Requirements** | Feature/bug branch ready to build with unit tests complete. |
| **Prerequisites** | Test code is integrated with the CI pipeline. |
| **Test Steps** | 1. Automated testing is triggered upon build tasks completing. |
| **Expected Output** | Application testing tasks run successfully and notifications are sent. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.9 Test Case: Application Analysis - Manual Execution

|  |  |
| --- | --- |
| **Description** | Verify the CI/CD pipeline application code analysis tasks run when the application code analysis feature is manually executed. |
| **Requirements** | Feature/bug branch ready to build and GitHub Actions configured. |
| **Prerequisites** | Code analysis tools are integrated with the CI pipeline. |
| **Test Steps** | 1. As DSO team member, manually kick off the application code analysis tasks. |
| **Expected Output** | Application code analysis tasks run successfully and notifications are sent. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.10 Test Case: Application Analysis - Automated Execution

|  |  |
| --- | --- |
| **Description** | Verify the CI/CD pipeline application code analysis tasks run after testing tasks are complete. |
| **Requirements** | Feature/bug branch ready to build and GitHub Actions configured. |
| **Prerequisites** | Code analysis tools are integrated with the CI pipeline. |
| **Test Steps** | 1. Automated code analysis is triggered upon testing tasks completing. |
| **Expected Output** | Application code analysis tasks run successfully and notifications are sent. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.11 Test Case: Triggers - Enable Continuous Integration (CI)

|  |  |
| --- | --- |
| **Description** | Verify when enabling continuous integration, the CI pipeline tasks run automatically when code changes are made to repository main branch. |
| **Requirements** | Feature/bug branch ready to build and merged into the main branch. |
| **Prerequisites** | The CI pipeline is integrated with the remote code repository. |
| **Test Steps** | 1. As DSO team member, turn on the continuous integration flag. 2. Merge integration branch into main branch. |
| **Expected Output** | Application build, test, and code analysis tasks run in sequence successfully and notifications are sent. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.12 Test Case: Triggers - Disable Continuous Integration (CI)

|  |  |
| --- | --- |
| **Description** | Verify when disabling continuous integration, the CI pipeline tasks do not run automatically when code changes are made to repository main branch. |
| **Requirements** | Feature/bug branch ready to build and merged into the main branch. |
| **Prerequisites** | The CI pipeline is integrated with the remote code repository. |
| **Test Steps** | 1. As DSO team member, turn off the continuous integration flag. 2. Merge integration branch into main branch. |
| **Expected Output** | Application build, test, and code analysis tasks do not run automatically. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.13 Test Case: Application Deployment - Manual Execution

|  |  |
| --- | --- |
| **Description** | Verify the CI/CD pipeline application deployment tasks run when the application deployment feature is manually executed. |
| **Requirements** | Feature/bug branch ready to build and deploy. |
| **Prerequisites** | Deployment environments are integrated with the CD pipeline. |
| **Test Steps** | 1. As DSO team member, manually kick off the application deployment tasks. |
| **Expected Output** | Application deployment tasks run successfully and notifications are sent. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.14 Test Case: Application Deployment - Automated Execution

|  |  |
| --- | --- |
| **Description** | Verify the CI/CD pipeline application deployment tasks run after code analysis tasks are complete. |
| **Requirements** | Feature/bug branch ready to build and GitHub Actions configured. |
| **Prerequisites** | Deployment environments are integrated with the CD pipeline. |
| **Test Steps** | 1. Automated application deployment is triggered upon code analysis tasks completing. |
| **Expected Output** | Application deployment tasks run successfully and notifications are sent. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.15 Test Case: Triggers - Enable Continuous Delivery (CD)

|  |  |
| --- | --- |
| **Description** | Verify when enabling continuous deployment, the CD pipeline tasks run automatically after CI tasks are completed. |
| **Requirements** | Feature/bug branch ready to build and GitHub Actions configured. |
| **Prerequisites** | Deployment environments are integrated with the CD pipeline. |
| **Test Steps** | 1. As DSO team member, turn on the continuous deployment flag. 2. Merge integration branch into main branch. |
| **Expected Output** | Application deployment tasks run successfully after CI tasks and notifications are sent. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

### 4.3.16 Test Case: Triggers - Disable Continuous Delivery (CD)

|  |  |
| --- | --- |
| **Description** | Verify when disabling continuous deployment, the CD pipeline tasks do not run automatically after CI tasks are completed. |
| **Requirements** | Feature/bug branch ready to build and GitHub Actions configured. |
| **Prerequisites** | Deployment environments are integrated with the CD pipeline. |
| **Test Steps** | 1. As DSO team member, turn off the continuous deployment flag. 2. Merge integration branch into main branch. |
| **Expected Output** | Application deployment tasks do not run automatically after CI tasks. |
| **Assumptions** |  |
| **Result** | **Pass** |

|  |  |
| --- | --- |
| **Failing Requirement** | *Test case that did* ***not work*** *as expected.* |
| **Actual Output** | *Explain the actual output you got and how it is different from expected (e.g. failing step or different output)* |
| **Defect** | *Explain why it is a defect or why it is not* |
| **Screenshot of Failure** |  |

## 4.4 Example Android Pipeline Test Results

|  |
| --- |
| Image |

## 4.5 Example iOS Pipeline Test Results

|  |
| --- |
| Image |