**Project Plan**

**Version 3.1**

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)** |
| 1/22/2022 | 1.0 | Initial document | Robert Wren  Andrew Nicolette |
| 02/06/2022 | 1.1 | Risk Matrix revision based on professor feedback and updated project schedule | Robert Wren  Andrew Nicolette |
| 03/05/2022 | 2.0 | Update for Milestone 2 | Robert Wren  Andrew Nicolette |
| 03/26/2022 | 3.0 | Add Software Test Plan for Milestone 3 | Robert Wren  Andrew Nicolette |
| 04/02/2022 | 3.1 | Final document | Robert Wren  Andrew Nicolette |

**Table of Contents**

1. Executive Summary 5

1.1 Statement of Need 5

1.2 Project Definition 5

1.3 Objectives 6

1.4 Scope 6

1.5 Definitions, Acronyms, and Abbreviations 7

1.6 References 7

2. Organization 8

2.1 Internal Structure 8

2.2 External Structure 8

2.3 Roles and Responsibilities 8

2.4 Communication Management Plan 10

2.5 Cost Analysis 12

3. Execution 12

3.1 Methodology 12

3.2 Key Deliverables and Milestones 12

3.3 Change Management 14

3.4 Assumptions and Constraints 15

3.5 Risk Analysis 15

4. Test Plan 16

4.1 Processes 16

4.2 Environment 17

4.2.1 Software Requirements 17

4.2.2 Hardware Requirements 17

4.3 Proposed Test Cases 17

4.3.1 Test Case: Version Control - Merge Development Branch (Developer) 19

4.3.2 Test Case: Version Control - Merge Development Branch (Lead Developer) 20

4.3.3 Test Case: Version Control - Merge main branch (Development Team) 21

4.3.4 Test Case: Version Control - Merge Main Branch (DSO) 22

4.3.5 Test Case: Application Build - Manual Execution 23

4.3.6 Test Case: Application Build - Automated Execution 24

4.3.7 Test Case: Application Test - Manual Execution 25

4.3.8 Test Case: Application Test - Automated Execution 26

4.3.9 Test Case: Application Analysis - Manual Execution 27

4.3.10 Test Case: Application Analysis - Automated Execution 28

4.3.11 Test Case: Triggers - Enable Continuous Integration (CI) 29

4.3.12 Test Case: Triggers - Disable Continuous Integration (CI) 30

4.3.13 Test Case: Application Deployment - Manual Execution 31

4.3.14 Test Case: Application Deployment - Automated Execution 32

4.3.15 Test Case: Triggers - Enable Continuous Delivery (CD) 33

4.3.16 Test Case: Triggers - Disable Continuous Delivery (CD) 34

4.4 Assumptions and Constraints 35

4.4.1 Assumptions 35

5. Future Work 36

6. Project Schedule 37

# 1. Executive Summary

## 1.1 Statement of Need

The Spring ’22 Software Engineering Capstone project requires the enhancement and development of a mobile application to assist people with short-term memory loss. This semester the goal will be to continue the work of prior teams and publish the app in the Apple App Store and Google Play. The project development team, United Global Masters Coders, took a vote and decided on the name, MemorEZ for the application. United Global Masters Coders will not be experimenting as much as in previous semesters, but will focus more on adding traditional aspects and productizing the app. To support the development of the MemorEZ application, the DevSecOps (DSO) team was formed. The DevSecOps Project Plan is the initial document to outline the schedule, techniques, and tools to support the fast-paced software development project with efficiency.

The MemorEZ DevSecOps project will enable the development teams to utilize a set of best practices and automation tools to build, test, secure, publish and deploy their services and applications. A continuous integration (CI) pipeline with a centralized code repository will be provided to the two development teams with the test automation and static code analysis tools to verify functionalities and ensure industry standards are followed, including security. A continuous delivery (CD) pipeline will also be provided to have the code built and deployed to the assigned environment.

## 1.2 Project Definition

The project aims to streamline software development and deployment via automated processes that may otherwise require manual intervention that can be time-consuming and/or error-prone. The process ensures a high-quality end product that can be deployed in a fast and reliable manner and on a consistent and frequent basis. In addition to development and deployment operations, security will also be emphasized as a requirement in code quality by utilizing security practices to identify vulnerabilities earlier in the feedback cycle. The CI/CD pipeline aims to ensure environment stability, enable high release velocity, reduce defects with automated testing, monitor processes, and provide metrics for tracking.

## 1.3 Objectives

DevSecOps aims to bring security directly into the software development lifecycle to attempt to avoid misconfigurations or weak implementations that can leave the app, service, and infrastructure vulnerable. DevSecOps emphasizes creating a security-aware culture based on collaboration across the entire team. It also urges teams to take a holistic view of the software lifecycle and consider platform solutions that make it easier to integrate security solutions. It argues that platform solutions offer the added benefit of enabling teams to identify inflection points in the application development process, where security controls can be inserted and automated.

Within the course context, the DevSecOps team exists to provide cohesion between the individual project teams and to serve as a unifying agent for the selection of industry-standard Tools, Techniques, and Practices (TTP). Guided by the Project Coordinator and Project Mentors, the DevSecOps team primary objectives are to:

* Develop an internal structure for software development across the segregated project teams
* Advise project teams through the respective Project Manager of best practices discovered through both external research and intra-team experiences
* Create standardized instructional documentation of each team’s project startup
* Provide technical design documents to aid in continuity across the team
* Lessons learned from this project will benefit future projects.

## 1.4 Scope

The guidelines of the Project Coordinator and, to the extent possible, the guidance of the DevSecOps team, are recommendations provided to the development teams representing an opinionated architecture structure for the development, testing, and delivery of each of the client’s requirement. Within the DevSecOps scope of operations, the team contributes to the term’s project execution by supporting the following tasks:

* + Create and manage the centralized code repository for each development team
  + Provide guidelines for industry standards and best practices for tech stacks
  + Support project development teams through setup and utilization of recommended development stacks
  + Provide CI/CD pipeline for each development to build, test, analyze and deploy.
  + Provide overall code reviews to establish and enforce an intra-team, agreed-upon code standards
  + Provide strategies to support CI/CD pipeline with limited resources with no cost solution

## 1.5 Definitions, Acronyms, and Abbreviations

The below list defines popular project terms and acronyms.

* CI/CD - Continuous integration and continuous deployment, or CI/CD pipeline, a set of operating principles, practices, and tools used to deliver code often and reliably.
* DSO - DevSecOps, Development, Security & Operations, or the set of practices that combines software development, IT security, and IT operations in the integration and deployment of software applications.
* DSO POC – DevSecOps point of contact from each team.
* MemorEZ - The application under development, a mobile application to assist people with short-term memory loss.
* Git – The project’s chosen version control system for source code.
* FlutteringMinds Team – The project development team, consisting of a PM, software developers, user interface/user experience (UI/UX) designers, software quality assurance engineers, and business analysts tasked with creating the mobile interface and services that the user would use to assist with short-term memory loss.
* PM – Project manager
* Repo - Source code repository to store code and documentation for software hosted on the web, such as GitHub.
* RememberAll Team – The second project development team, consisting of a PM, software developers, user interface/user experience (UI/UX) designers, software quality assurance engineers, and business analysts tasked with creating the mobile interface and services that the user would use to assist with short-term memory loss.

## 1.6 References

Pham, I., Leung, V. (2021, February 02). DevSecOps Project Plan. University of

Maryland Global Campus.

# 2. Organization

The MemorEZ project consists of an overall project manager, DSO team, and two development teams. Each application team has a project manager, developers, business analysts, and testers. All teams are led by the project coordinator and project sponsors.

## 2.1 Internal Structure

As the DSO team has only two members, each member shares all roles and responsibilities to ensure the continuity of the project. DSO team collaborates within itself throughout the life of the project to provide consistent recommendations and best practices to all application teams.

## 2.2 External Structure

The DSO team’s main responsibility is to serve all teams where it is deemed necessary. Therefore, the DSO team can be reached via Teams and email from any stakeholders to provide technical assistance and DSO guidelines. The DSO team will also reach out to stakeholders as needed per the communication plan. While the DevSecOps team aims to provide technical assistance to all project team members, the primary external interface between each of the development teams is through or in coordination with the respective Project Manager and Lead Developer/DSO POC for any given team. This ensures that any DevSecOps requests or decisions have the vetting of all team members, and between project PMs when appropriate.

## 2.3 Roles and Responsibilities

DevSecOps brings the security team into all phases of the software development process and emphasizes creating a security-aware culture based on collaboration across the entire team. This collaboration is key to ensuring security issues are addressed as early as possible, where they are easiest and least expensive to address.

DevSecOps is about delivering secure services as quickly as possible. While DevOps has conflicting goals and strategies between the DevOps and security teams, DevSecOps creates a security-aware culture based on collaboration across the entire team from the beginning of the process.

The responsibilities of the DSO team include, but are not limited to, the planning, designing, and implementation of a continuous integration/continuous deployment pipeline to support the development and deployment of the MemorEZ application for the Capstone Project. The team members will work together to document and implement a working CI/CD pipeline for eventual deployment of said application under development, as well as be liaison and support for development team members, its customers. The DSO team will set and uphold such things as best practices, coding standards, and offer technical expertise and assistance to development teams using its framework, in addition to soliciting feedback and suggestions for improvement.

At the time of this document inception, from a DevSecOps perspective, there are several key roles that are recognized and assigned to key Capstone participants:

* + Project Sponsor: Dr. Mir Mohammed Assadullah is currently the sole proprietary of this role and serves as chief advisor of all projects. As such, Dr. Assadullah may exercise authority to alter, limit, or expand any of the perimeters of the project(s) to meet Capstone requirements.
  + Project Mentors: The experts in the information technology who provide guidance and advisory for any impediments, issues and improvements.
  + Project Managers: There is an overall project manager who works closely with team project managers to track the timeline and deliverables. Team PMs are liaisons between their assigned team and all stakeholders. The PM communicates any needs or concerns from the teams to affected stakeholders and vice versa. Each team has identified a primary Project Manager; however, to allow for flexibility in team roles, teams may collectively decide to reassign the role during any particular phases of the project. Such a change must adhere to the communication plan to notify the core steering committee members.
  + Software Developers: Identified developers are responsible for utilizing Capstone-specified programming languages, tools, and infrastructure to write, test, and implement client specified applications. At times during the production effort, it may be necessary for DSO team members to interface directly with a given project's assigned developers outside of the specified communication plan to establish processes, clarify requirements, or perform similar related items to meet the overall Capstone goals.
  + Steering Committee: Refers to those critical roles needed to make the majority of critical decisions that result in an impact on multiple team's internal processes or deliverables. As of this writing, those roles are:
    - Project Coordinator
    - Project Mentor
    - Overall Project Manager
    - Team Project Manager
    - DevSecOps

## 2.4 Communication Management Plan

The DevSecOps team will strive to communicate frequently to all project stakeholders and respond in a timely manner to requests and incidents. The DSO project status will be tracked and updated frequently to ensure progress towards milestone goals and fulfillment of deliverables. It is preferable that development and project management team members message DSO members through the project’s Microsoft Teams channels and chat sessions. However, other options are also available such as email or phone. The DSO team will liaison with individual development teams via points of contact of respective teams as noted as Lead developer/DSO POC. The DSO team will escalate issues to team project managers and business management as needed.

**Approach**

With interface requirements for each of the project teams, each DSO team member leads the effort of ensuring effective communication within the Capstone construct. They communicate with the steering committee members to determine their preferred frequency and method of communication. At the current time, the steering committee is notified by weekly update emails to ensure projects on individual schedules. In addition to identifying communication frequency and method, the project's communication channels also identify and confirm that stakeholders have access to respective channels. After changes get approved, the plan is updated, and supporting documentation is distributed to update the project team and all stakeholders.

**Project Team Directory**

**Project Advisor/Mentor**

|  |  |  |
| --- | --- | --- |
| **Role** | **Team Member** | **Contact** |
| Project Advisor | Dr. Mir Mohammed Assadullah | mir.assadullah@faculty.umgc.edu |
| Project Mentor (DevSecOps/Cloud) | Robert Wilson | rwilson@umgc.dev |
| Project Mentor (BA/PM/Documentation) | Roy Gordon | uspsrgordon@aol.com |
| Project Mentor (Technology Architecture) | Daniel Avery | daniel@danielavery.net |

**Overall Project Management**

|  |  |  |
| --- | --- | --- |
| **Role** | **Team Member** | **Contact** |
| Project Manager | James Eble | jve2nd@gmail.com |

**DSO Team**

|  |  |  |
| --- | --- | --- |
| **Role** | **Team Member** | **Contact** |
| DevSecOps Engineer | Robert Wren | rwren@student.umgc.edu |
| DevSecOps Engineer | Andrew Nicolette | nicolettedrew0@gmail.com |

**FlutteringMinds Team**

|  |  |  |
| --- | --- | --- |
| **Role** | **Team Member** | **Contact** |
| Project Manager | Selina Zaman | selinazn25@gmail.com |
| Lead Developer/DSO POC | Daryle Urrea | daryle.urrea@hotmail.com |
| Developer | Joshua Fischer | joshuadf@ymail.com |
| Developer | Anusha Ramanan | aramanan1@student.umgc.edu |
| Business Analyst | Vanessa Stringer | vmbstringer@gmail.com |
| Business Analyst | Sean LaMonica | seanlamonica@gmail.com |
| Test Engineer | Joseph Jewell | jjewell1989@gmail.com |

**RememberAll Team**

|  |  |  |
| --- | --- | --- |
| **Role** | **Team Member** | **Contact** |
| Project Manager | Brian Avadikian | bavadikian@gmail.com |
| Lead Developer/DSO POC | Yusufu Sanu | yusufusanu@gmail.com |
| Developer | Vivek Singh | vivek343@gmail.com |
| Developer | Lizset Chavez Chacaltana | lizset\_p2004@hotmail.com |
| Developer | Johnnie Webb | johnniewebb@usa.net |
| Business Analyst | Genet Asmelash Medhanie | genetmedhanie@gmail.com |
| Business Analyst | Eyob Woldehana | eyobzz@yahoo.com |
| Test Engineer | Robert Edwards | rob\_ee@hotmail.com |

## 2.5 Cost Analysis

It is the mission of the DevSecOps team to provide all project teams and the Project-Sponsors little to no cost solutions to on which to develop, deliver, and prototype software deliverables. Within these boundaries, the expected cost of all project-related expenditures is non-existence to negligible. Providing a no-cost solution limits the capability of the deliverable products to operate in the realm of open-source, freely available platforms. If accepted, the Project Sponsor may expand the capabilities of the specified deliverables at their discretion; however, they accept all cost increase from the expansions of said products.

While the project teams may exercise their independence and utilize any for-cost products in their internal development operations, the team cannot pass such expenses on to the client. Lastly, after the Capstone specified timeline, the teams must offer their deliverables in a non-expenditure acquired deployment solution.

# 3. Execution

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 Methodology

The Capstone project is organized in a 12-week session starting January 8, 2022 with a project kickoff and a four-phase approach identified by Milestones 1 through 4. With this specification, each phase has a definitive start/stop marker, with measurable criteria for success (identified by team-specified deliverables) that closes before the next phase. However, acknowledging the dynamic nature and scope of each software development initiative, changes or decisions from earlier phases may be revisited. Nevertheless, this is done within firm adherence to the specified Change Management Plan (CGMP). The included project schedule provides an outline of each of the major phases, with expected milestones.

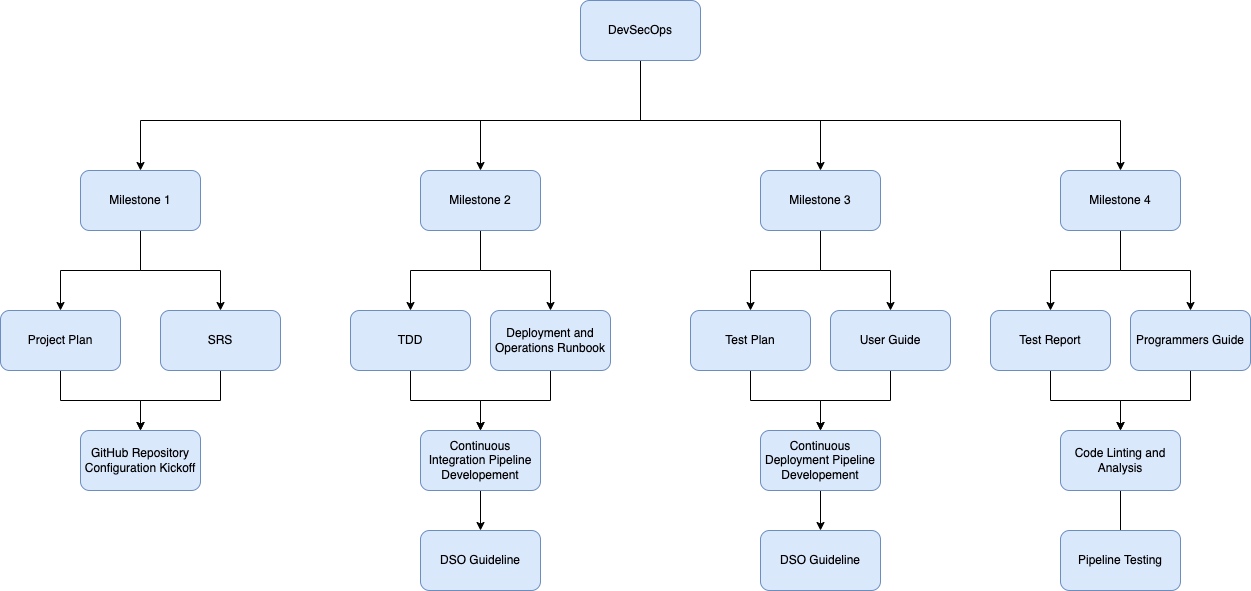
## 3.2 Key Deliverables and Milestones

The MemorEZ DevSecOps Project schedule shall take place over the course of eleven weeks and be delineated by four milestones to coincide with those of the development teams and overall project timeline. The milestone name, due date, and deliverables are detailed in the DevSecOps Milestones and Deliverables table below. Refer to descriptions below for details on contents of said deliverables.

**DevSecOps Milestones and Deliverables**

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Due Date** | **Deliverables** |
| One (1) | 01/22/2022 | Project Plan, SRS |
| Two (2) | 02/06/2022 | TDD, DSO Runbook |
| Three (3) | 03/26/2022 | Test Plan, User Guide |
| Four (4) | 04/2/2022 | Test Report, Programmers Guide |

* Project Plan – This document, detailing the overall plan, schedule, and deliverables of the MemorEZ DevSecOps Project. The document will be updated as required during the life of the project. However, it will be updated with a major version for each milestone.
* SRS - Software Requirements Specification document, describing the requirements for the CI/CD pipeline supporting the development and deployment of the MemorEZ application.
* TDD - Technical Design Document, describing the design of the entire CI/CD pipeline and its individual components.
* DSO Runbook - Deployment and Operations Guide (Runbook) document, an operation manual with steps on executing processes related to set up and configuration of the DSO pipeline.
* Test Plan - Document detailing the scope and strategy for testing the implementation of the DSO pipeline which will be added to the Project Plan at milestone three.
* User Guide - Document detailing the best practices, expectations, and required tools and environments to get started on integration, testing, and deployment of the project application.
* Test Report - Document detailing the test execution findings for testing session(s) on the DSO pipeline.
* Programmers Guide - Document detailing the implementation of the pipeline as reference for future DevSecOps teams for usage or maintenance.

**Figure 1:**  **Work Breakdown Structure.** WBS for proposed DevSecOps structure. 

## 3.3 Change Management

Change Management, in this context, refers to the changes that deviate from the outline in this document. Whether changes are trivial or complicated, the project team must manage changes to prevent scope creep. The CGMP includes submitting, reviewing, and approving changes at all echelons of the development process. All requests and submissions follow the change control process described below to ensure changes are approved, tracked, and managed per this CGMP.

The proposed CGMP approach confirms that changes throughout the project lifetime are defined, reviewed, and agreed on and adequately implemented and communicated to all involved parties. Also, it ensures that only changes within the scope of the project are considered. By following these processes, it is the goal of the DevSecOps team to prevent unnecessary changes from occurring and help the team members to focus only on beneficial changes within the project scope.

* + Internal change(s) to the DevSecOps project shall be communicated from one DSO member to the other via Teams chat, email, or phone. If the change(s) affect the development teams, the information shall be passed on to development team DSO POCs and development team PMs via a link to the DSO Teams wiki page or a document within the DSO Teams file repository.
  + External change(s) to the MemorEZ project that require a new design or implementation of DSO shall require consultation with both DSO members, development team PMs, overall PM, and optionally, development team DSO POCs.
  + For communications outreach, the DSO team shall contact project team members in the order below, with each line following representing an additional level of escalation:
    - Development team(s) as a whole
    - Development team DSO POCs
    - Development team PMs
    - Overall PM
    - Project advisors
    - Project coordinator

## 3.4 Assumptions and Constraints

To enable each project team the freedom to operate as independent entities while also meeting the expectations levied by the Project Organizer, DevSecOps operates under the following assumptions and constraints:

•[Assumption] Timetables: As mentioned earlier, any provided timetable and schedules are heavily influenced by the ability of individual teams to formulate and deliver software.

•[Assumption] Production Value: All software architectures and implementation are designed based on providing proof-of-concept and product prototyping.

•[Constraint] Architecture Requirements: Unless prior coordinated with the Project Coordinator, the selection of specific technologies is limited to this expressed in earlier requirements documents.

•[Constraint] Capstone Resources: Unless otherwise offered by the Project Sponsors, the Capstone provides no financial support for development efforts.

## 3.5 Risk Analysis

Identified risks run the gamut from internal, external, to project management related risks. Some of the major risks MemorEZ DevSecOps identified relate to potential shortcomings in inter-team and intra-team communications because of the nature of the development environment, assumptions of familiarity with and reliance on tools and documentation, and possible difficulties during components integration. The table below summarizes these and other risks, ranked by probability and impact.

DevSecOps Risk Probability and Impact Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Impact** | **High** | Technical skills deficiencies | CI/CD pipeline quality and usability issues | Issues during integration of components of MemorEZ |
| **Medium** | Development time underestimated | At least one DSO team member becomes unavailable | Pipeline features not sufficient to meet developer needs |
| **Low** | Over-reliance on tools and documentation over people | Project change management processes are not followed internally and/or externally | Communications to and/or from dev teams or PM may be incomplete or untimely |
| **Low** | **Medium** | **High** |
| **Probability** | | | | |

# 4. Test Plan

The Purpose of this document is to describe the processes, environment, and activities for

validation and verification of MemorEZ.

The Test Plan is designed to prescribe the scope, approach, resources, and general process of all testing activities of the project MemorEZ.

The plan identifies the items to be tested, the features to be tested, the types of testing to

be performed, the personnel responsible for testing, the resources and schedule required

to complete testing, and the risks associated with the plan.

## 4.1 Processes

Automated testing of the CI/CD pipeline features shall be included. Due to the pre-

defined nature of a DevSecOps pipeline, test coverage is limited to the core

functionalities. Development operations processes run by the pipeline falls under the

testing scope. The tester does not have access to nor needs knowledge of the internal code logic of the git implementation of GitHub, the build and deploy pipelines of GitHub

Actions, and those of the container and container orchestration implementations as it is

not these that are under test. Verification of the pipeline functionality is to ensure output

matches expected results. In-scope components include actions such as building, testing,

scanning, and deployment of the MemorEZ components. Out-of-scope components

include, and are not limited to, testing activities such as security or penetration testing,

localization testing, or usability testing.

The Spring 2022 development teams have developed the test plan in their Project Plan

documentation which focuses on UI (User Interface) and end to end testing for the

application. Most test cases are being done manually. The development teams have

planned to write some unit testing into their solutions. DSO will work with development

teams to ensure the pipeline runs and checks for unit tests with code coverage. However,

DSO is not responsible to create unit test cases in the solution. DSO will rely on the

development teams testers to write the unit testing in the solution so that the pipeline can

run directly from the solution.

## 4.2 Environment

### 4.2.1 Software Requirements

The test environment shall consist of a desktop operating system with a modern web browser. The CI/CD pipeline components shall be accessed via aforementioned browser. Testing shall be done on macOS Monterey 12.3 and Chrome browser Version 99.0.4844.83.

### 4.2.2 Hardware Requirements

Testing shall be conducted on a MacBook Pro laptop computer, model M1 Max. Keyboard input shall be done with the built-in keyboard and other inputs shall be done with the built-in trackpad.

## 4.3 Proposed Test Cases

Automated testing will be the main form of testing for this cycle. Due to the nature of the

pipeline, which is dependent on development team inputs in the form of source code, and

by which the source code is also dependent on said pipeline to build, test, and deploy it,

testing is conducted when pipeline run(s) are initiated, either manually or automatically.

Test cases will be designed to represent use-case scenarios as by DevSecOps personnel

and other development project stakeholders. The test suite shall be validated to ensure

that coverage is satisfactory for features and possible scenarios including positive and

negative cases when possible, and permutations of selectable options. The actual results

shall be compared to expected results to determine test case outcome in a later Test

Report. Development teams will include in their testing requirements unit and end-to-end

tests.

### 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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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** |  |

|  |  |
| --- | --- |
| **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 Assumptions and Constraints

### 4.4.1 Assumptions

Testing assumes that the test environment is available at the time of execution. It is assumed that all the development teams will provide functioning source code for which the CI/CD pipeline is dependent upon. It is assumed that the providers of the chosen DevSecOps tools shall not change usage allowances to free and student tier accounts during the duration of the project. The requirements stipulated in the MemorEZ DevSecOps Software Requirements Specification document shall represent the requirements under which the pipeline shall be tested and not change for the duration of the test cycle.

4.4.2 Constraints

The test lifecycle shall take place within the latter half to quarter of the UMGC Capstone, coinciding with other development team activities, encompassing just a few weeks/sessions with uncontrolled variables and compatibility. There shall be no traditional software testing lifecycle or bug tracking/fixing processes because of the nature of DevSecOps projects as infrastructure services.

# 5. Future Work

The below items have been identified as possible additions or improvements to guide

future development phases for the MemorEZ DevSecOps project.

* Automate application process for Google Play
* Automate application process for App Store

# 6. Project Schedule

**Figure 2:** Gantt chart for proposed DevSecOps tasks.

|  |
| --- |
| Image |
| Image |