Software Requirements Specification

for

FormBot DevSecOps

Version 2.0

Prepared by Ivy Pham & Vincent Leung

UMGC SWEN670 – Spring 2021

02/25/2021

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Features 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 3

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3. System Features 4

3.1 Remote Code Repository 4

3.2 Automated Application Build 4

3.3 Automated Application Testing 5

3.4 Automated Code Analysis 5

3.5 Automate Application Deployment 5

4. External Interface Requirements 5

4.1 User Interfaces Overview 5

4.2 Hardware Interfaces 5

4.3 Software Interfaces 5

5. System Features/Modules 6

5.1 Remote Code Repository 6

5.2 Automated Application Build 6

5.3 Automated Application Testing 7

5.4 Automated Code Analysis 7

5.5 Automated Application Deployment 8

5.6 Expansion of ADF Framework 8

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Vincent Leung | 1/25/2021 | Initial version | 1.0 |
| Vincent Leung | 1/25/2021 | Added sections 1 and 2 | 1.1 |
| Vincent Leung | 1/27/2021 | Added section 3 | 1.2 |
| Ivy Pham | 2/1/2021 | Various edits and additions | 1.3 |
| Ivy Pham | 2/2/2021 | Various edits and additions | 1.4 |
| Vincent Leung | 2/3/2021 | Final Review (Milestone 1) | 1.5 |
| Ivy Pham | 2/4/2021 | Final Review (Milestone 1) | 1.6 |
| Vincent Leung | 2/15/2021 | Incorporated changes from post Milestone 1 | 1.7 |
| Ivy Pham | 2/25/2021 | Review and Submit for 2nd milestone | 2.0 |
| Ivy Pham | 3/30/2021 | Final Review | 3.0 |

# Introduction

## Purpose

This Software Requirements Specification document describes functional and nonfunctional requirements for the Form Scriber DevSecOps project. This document is to be of use for development team members and other stakeholders to reference the conception of a continuous integration and continuous deployment pipeline, an automated framework used to build, test and secure, and deploy applications to their environments, primarily conducted on their personal development machines. Features described relate to the core functionality of the pipeline and are committed to version 1.0.

## Document Conventions

This SRS shall consist of six sections and follow IEEE 830 conventions. Sections are separated by major numbers, which are decomposed into subsections with minor numbers. Font size and styling shall follow this hierarchy accordingly.

## Intended Audience and Reading Suggestions

The SRS is to be accessible to all internal and external team members within the product group. End users shall refer to later documentation such as the operations manual (runbook), programmer’s guide, and/or readme files. If further information about project conception is needed, the user may request access to working documentation such as this SRS via contact with project sponsors. This document is designed to be read sequentially and specific team members shall find some sections more pertinent than others. For example, project management and business analysts shall find sections 1 and 2 (Introduction and Description) more useful, whereas sections 3, 4, and 5 (Features and Interfaces) more useful for developers and testers. Overlap of these sections is expected as project roles are expected to be fluid.

## Product Scope

The Software Engineering Capstone project requires the creation of a prototype application called Form Scriber, which is the development of an artificial intelligence (AI) software to support doctors, medical technicians, police officers and others. The application primarily assists professionals in public service to capture and prepare text reports during conversations with their clients. For example, the conversation between a physician and patient can be recorded to automatically prepare reports that can be edited and saved to an Electronic Medical Record (EMR). To support the development of The Form Scriber application, the DevSecOps (DSO) team was formed.

The Form Scriber DevSecOps project focuses on the creation of an infrastructure to automatically build, test, verify security, and deploy the three components that comprise the complete Form Forscriber application in a consistent and reliable manner, easing burdens on development teams of manual processes and checks. An effort to include container creation and orchestration shall be included to address theoretical needs for scaling, availability, and maintainability during and after application deployment. The project and team shall act as facilitators of tools and processes that enable individual development teams to realize their products and use the DSO pipeline to monitor for correct integration with other Form Scriber components. In this respect, the DSO project aims to provide a self-service “service” and ongoing support to team members to achieve such goal rather than act as code owners.

# Overall Description

## Product Perspective

The Form Scriber DSO project is a complement to the development methodology chosen, providing services and technologies to development teams that enable rapid delivery of high-quality software on a frequent basis. In existing organizations, this may mean a replacement for existing manual processes, however for Form Scriber, development of the DSO pipeline is established in parallel to and in collaboration with development teams. As there is no traditional software model with dozens or hundreds of scenarios and use cases, the SRS shall concentrate on a few key areas such as providing apparatuses to ensure a stable environment, to successfully create a build of the application, to successfully test and ensure compliance with security guidelines, and to successfully deploy said application.

## Product Features

The DSO CI/CD pipeline features tools and technologies that shall allow users to do the following:

• Manage and track changes to source code

• Enable team collaboration with the remote code repository

• Run automated tests on builds

• Scan for code quality and security vulnerabilities

• Deploy to resources

## User Classes and Characteristics

**Developer** – This user is the highest priority for support and represents the main class of user of the DSO pipeline. Developers shall use the technologies provided within the context of the DSO mindset to facilitate integration and deployment activities. Within this class is a subset member, the DSO POC (point of contact), who shall be designated as the member who would liaison with DSO team members to merge development branches of the code repository into the main or master branch, among other development operations tasks and requests.

**DSO POC –** This user is the main point of contact from each development team for any related DSO matters. As described above, DSO POC may have multiple roles in a team such as the lead developer or developer. DSO POC can do the peer review and ensure DSO guidelines are followed before merging changes to the development branch.

**Admin** – This user is part of the Form Scriber DevSecOps team. The admin selects, configures, installs, manages, and maintains all the various components of the CI/CD infrastructure appropriate for the development requirements. The admin user has the highest privileges in regards to maintenance of the pipeline, but is not considered the owner(s) of the content for which the project enables delivery.

**End user** – The end user does not have access to the instances used by team members during development, but has full access to the final product delivered and license to implement it or their own versions of it however they see fit. Access to the content would be through the GitHub repository.

## Operating Environment

The following are operating environment considerations:

OE-1: The DSO project shall support multiple operating systems as developers use their personal machines with innumerable configurations. The project utilizes industry tools that have clients for desktop systems such as Windows, MacOS, and Linux. Support shall be with development machines that are of relatively recent vintage.

OE-2: Much of the access to resources shall be via the Internet through modern web browsers such as Chrome, Firefox, and Safari.

OE-3: Feedback is solicited from development teams for the coding languages and frameworks that shall be used so that DevSecOps team could design the pipeline while proactively considering these requirements.

OE-4: For cost effectiveness, the DSO project shall leverage containerization tools such as Docker where it is applicable to deploy and test components of the application.

## Design and Implementation Constraints

The following are design and implementation constraints:

CO-1: Dialogflow implementations shall utilize Go and SQL.

CO-2: Mobile implementations shall utilize Dart/Flutter, JavaScript and HTML5

CO-3: DSO pipeline shall utilize Azure services and third-party tools with no associated cost.

## User Documentation

The following documentation shall be delivered with the software:

UD-1: Deployment and Operations Guide (Runbook)

UD-2: Programmer’s Guide

UD-3: User Guide

## Assumptions and Dependencies

Successful development of the CI/CD pipeline is contingent on a number of assumptions and dependencies:

AS-1: Continued free access to the remote code repository.

AS-2: Allotment of free $200 credit in Azure student subscriptions is sufficient for implementation.

AS-3: Development team members have a robust and stable high-speed Internet connection.

AS-4: DSO POC from each development team is available for consultation during implementation.

# System Features

## Remote Code Repository

The feature utilizes the free tier of a distributed version-control code repository for tracking and collaboration which allows developers to retrieve and contribute to their project’s source code in collaboration with team members. The feature also provides the protected branch feature which allows DSO team and developers to control changes to specific branches.

## Automated Application Build

This feature allows the DSO team to work with developers to add the build solution for each component of the application to the DSO pipeline where the pipeline is integrated with the remote code repository and triggered by new merges. The feature also allows the pipeline to automatically compile the code according to specific programming languages. The notification is sent to the appropriate team for the status of each build. The build task can also be run manually as needed.

## Automated Application Testing

This feature allows DSO team to work with developers to add the testing solution for each component of the application to the DSO pipeline. It is triggered after the build is successful. The feature also allows the pipeline to automatically run the selected functional testing (Unit/Integration). The notification is sent to the appropriate team for the status of each run. The testing task can also be run manually as needed.

## Automated Code Analysis

This feature allows DSO team to conduct the code quality and security checks. The feature allows the pipeline to automatically analyze the code to agreed-upon specifications and generate reports for remediations before the deployment. The code analysis task can also be run manually as needed.

## Automate Application Deployment

This feature allows the DSO team to release the application to an environment for various purposes after the code has passed the build, test, and code analysis. The deployment task can also be run manually.

# External Interface Requirements

## User Interfaces Overview

As it was determined from the inception for cost effectiveness, Microsoft Azure services, Azure DevOps tools, GitHub and Docker.IO will be utilized, which provide both graphical user interfaces and command-line methods for management and usage.

## Hardware Interfaces

No specific hardware interfaces have been identified for the system. DevSecOps team shall utilize free Azure services and third-party tools for cost effectiveness. Development teams shall utilize their personal machines with access to the Internet for interfacing with the DevSecOps pipeline in the cloud. Development teams shall also use their Microsoft accounts to receive a free $200 credit for Azure services where applicable such as building containers to deploy and test their components.

## Software Interfaces

SI-1: Various DevSecOps tools shall utilize a command line or user interface client either native to the operating system or an equivalent third party.

SI-2: Notification of process activities shall be sent via email communications to configured external email addresses.

# System Features/Modules

## Remote Code Repository

5.1.1 Description

A developer who has met all requirements (OE, CO, AS and SI) including guidelines from DSO to clone the remote code repository to the local computer for development collaboration. The developer creates the working/temporary branch such as a feature branch and makes changes to the code. The developer pushes the new branch to the remote code repository. The developer requests to merge changes to the protected branch such as the development or master branch.

5.1.2 Stimulus/Response Sequences

Stimulus: A developer requests to merge changes to a protected branch.

Response: The system displays the restriction of the protected branch with rules and the information to merge. The lead developer/DSO POC or a DSO team member (the approving authority) who are configured to have the permission to merge is also notified via email.

Stimulus: The approving authority approves the merge.

Response: The system merges new changes to the protected branch and tracks those changes. The system also notifies the requester for the approval or rejection.

5.1.3 Functional Requirements

REQ-1.1: Upon a developer requesting to merge to the projected branch, the system shall (a) display rules and information of the approving authority, and (b) send the merge request notification to the configured approving authority.

REQ-1.2: Upon the approving authority approving the merge request, the system shall (a) merge the changes to the protected branch, (b) track those changes, and (c) notify the merge requestor.

## Automated Application Build

5.2.1 Description

After changes are merged to the protected branch, the DSO pipeline, which is integrated with the remote code repository and is monitoring the protected branch, is triggered by the new changes from the merge. The pipeline automatically runs the build process to ensure the solution/application is built successfully based on the specifications of the chosen programming languages and tools.

5.2.2 Stimulus/Response Sequences

Stimulus: Changes are merged to the protected branch approved by the approving authority.

Response: The pipeline detects changes in the protected branch and triggers the task to build the application/solution. The pipeline also notifies the approving authority and the merge requestor of the success or failure status of the build.

5.2.3 Functional Requirements

REQ-2.1: Upon changes merging to the protected branch, the pipeline shall (a) be integrated with the remote code repository, (b) monitor changes in the protected branch, (c) trigger the build task of the application/solution, and (d) notify the approving authority and the merge requestor of the success or failure status of the build.

## Automated Application Testing

5.3.1 Description

After the build task is run successfully, the DSO pipeline triggers selected functional testing task (Unit/Integration) as agreed upon by the team.

5.3.2 Stimulus/Response Sequences

Stimulus: The build task is successfully run and triggers the selected functional testing task.

Response: The pipeline runs the selected functional testing task. The pipeline also notifies the approving authority and the merge requestor of the status and generates a report.

5.3.3 Functional Requirements

REQ-3.1: Upon the build task running successfully, the pipeline shall (a) trigger the selected testing task (Unit/Integration), and (b) notify the approving authority and the merge requestor of the status with the generated report.

## Automated Code Analysis

5.4.1 Description

After the testing task is completed, the DSO pipeline triggers the code analysis task to check for the code quality and security for remediation as needed.

5.4.2 Stimulus/Response Sequences

Stimulus: The testing task is completed and triggers the code analysis task.

Response: The pipeline runs the code analysis task. The pipeline also notifies the approving authority and the merge requestor of the status with generated report.

5.4.3 Functional Requirements

REQ-4.1: Upon the testing task completing, the pipeline shall (a) trigger the code analysis task and (b) notify the approving authority and the merge requestor of the status and generated report.

## Automated Application Deployment

5.5.1 Description

After the code analysis task is completed, the DSO pipeline triggers the deployment task to configure and install the application/solution to a selected environment.

5.5.2 Stimulus/Response Sequences

Stimulus: The code analysis task is completed and triggers the deployment task.

Response: The pipeline runs the deployment task. The pipeline also notifies the approving authority and the merge requestor of the status.

5.5.3 Functional Requirements

REQ-5.1: Upon the code analysis task completing, the pipeline shall (a) be configured to deploy to a selected environment, (b) trigger the deployment task, (c) notify the approving authority and the merge requestor of the status, and (d) utilize a container orchestration strategy and its associated tools.

## Expansion of ADF Framework

5.5.1 Description

A research and development effort has been elected to expand a previously-created DSO framework to include support for another development language that could theoretically be used by a FormBot development team.

5.5.2 Stimulus/Response Sequences

Stimulus: The user installs the new expansion for the ADF framework.

Response: The framework installs pre-determined dependencies and development environment for the language the expansion supports.

5.5.3 Functional Requirements

REQ-5.1: Upon the user cloning and running the ADF code, the appropriate development environment variables and dependencies are set up in order to enable application builds via containerization and deployment to local machines.