Software Requirements Specification

for

Advanced Development Factory for Flutter / Dart

Version 1.5

Prepared by Jeroen Soeurt, Michelle Monfort, and Robert Wilson

UMGC SWEN 670 Summer 2021

June 18, 2021

Table of Contents

Revision History

Name	Date	Reason for Changes	Version
Jeroen Soeurt	*June 5, 2021*	*Initial version*	*1.0*
Rob Wilson	*June 5, 2021*	*Review*	*1.1*
Michelle Monfort	*June 5, 2021*	*Review*	*1.2*
Rob Wilson	*June 11, 2021*	*Final*	*1.3*
Jeroen Soeurt	*June 18, 2021*	*Conversion to IEEE 830*	*1.4*
Michelle Monfort	*June 19, 2021*	*Review*	*1.5*
Rob Wilson	*June 20, 2021*	*Review*	*1.6*

1. Introduction

1.1 Purpose

The Software Reference Specification (SRS) document outlines key specifications for the Advanced Development Factory (ADF) for the development of the UMGC proposed application. The document specifically targets the key specifications of the software and how the Development Security Operations (DevSecOps) team will be incorporated in the facilitation of the software development teams building the application. The specifications for these teams will be generalized so that software can be modified and implemented to meet the specific requirements for any agile-hybrid development style. Integrated Development Environment use will be discussed, and software suggestions will be delivered. The purpose of this document is to provide an overall structure and guidance for common problems that occur when building software as well as provide a basis for a standardized development environment for the development teams.

The first chapter of this document provides an overview of the Advanced Develop Factory that we propose. Since this is a technical product, jargon will often we used. Chapter 2 describes the overall product description. Chapter 3 provides a brief overview of the features. User, hardware, and software interfaces are described in chapter 4. Chapter 5 lists the functional requirements for this product. Chapter 6 lists the non-functional requirements for this product.

1.2 Document Conventions

The Advance Development Factory aims to provide a consistent and reproducible environment for learners to reduce the distraction of setting up the development environment. The perspective and goals of the project are described in detail in section 2.

1.3 Intended Audience and Reading Suggestions

This document is for professor Dr. Assadullah, project mentor Johnny Lockhart, and members of the DevSecOps team. The overall verbiage of the document is intended for seasoned software engineers.

1.4 Product Scope

The ADF provides a standard development environment for the students of UMGC. It consists of a Docker image containing all required software to run, test and build the application, and an instruction manual.
The goal is to make the system platform agnostic. Requiring that the user only needs access to the internet in order to work within the required development environment and a local installation of Microsoft Visual Studio Code. Microsoft Visual Studio Code, an open-source product and cross-platform product, is the intended IDE to use in combination with this Docker image. This IDE can develop inside a Docker image using the Remote SSH plugin. It also offers wide support for

varying languages including Java, C++, C#. .Net Core, T-SQL, Go, Python, and Dart for example (Microsoft, 2021). There are thousands of extensions for supporting a wide variety of develop environments.

1.5 References

Docker Inc. (n.d.). *Docker Docs*. Retrieved from Install Docker Engine: https://docs.docker.com/engine/install/

Microsoft. (2021). *Languages Overview*. Retrieved from Visual Studio Code: https://code.visualstudio.com/docs/languages/overview

Microsoft. (n.d.). *Visual Studio Code*. Retrieved from https://code.visualstudio.com/

SonarCloud. (n.d.). *SonarCloud Home Page*. Retrieved from https://sonarcloud.io

2. Overall Description

2.1 Product Perspective

The Master of Information Systems with a Specialization in Software Engineering program at the University of Maryland Global Campus ends with a capstone course. In this course, the students are divided over several development teams, and a DevSecOps team. UMGC is divided into a stateside, a European, and an Asian division, and has tens of thousands of students all over the world. Many of their students are full-time employed in the Military or the civilian sector. A direct result of this is that development teams are geographically separated. A standardized development environment is more difficult to fulfil. The ADF that we propose will provide that standardized development environment for the development teams.

Several teams have worked on a similar product in previous semesters, but often they were targeted at a specific programming language. They also only provided command line tools. We intend to build on their ideas, as well as add support for multiple programming languages, and a graphical user interface containing an integrated development environment.

2.2 Product Features

Product features consist of the following bulleted points:

- Conduct GIT Operations
- IDE Support Project Type
- IDE Supports IntelliSense
- Build Support
- Automated Test Support
- Remotely Accessible
- Code Analysis Support

2.3 User Classes and Characteristics

This product is intended to be used by members of the development teams in the UMGC SWEN 670 capstone course. These teams consist of the follow user classes:

Developer
Users in this class will be the primary users of the system. They use it to develop software, and will use all features described in 2.2

Tester
Testers test the software that is written by developers. They use most features of the product, as they will pull repositories, build the product, and write and run tests. They also commit their unit tests to the repository.

Business Analyst
This group of users analyzes the problem that is being solved, and research approaches, techniques, and solutions. They do not write any code and will likely have limited use for the ADF.

Project Manager
The project manager manages the overall project and will not write code. They will have limited use for the ADF.

Caveat: *The SWEN 670 capstone course may require that every team member might step out of their specific role in the project team and assist in development. As such, any member might have the same needs as the developer user class.*

2.4 Operating Environment

1. The ADF will run in a Docker container, either in the cloud or on an end-user's computer. The end users connect to the graphical user interface using the Remote Desktop Protocol. This means that if the user wants to run the Docker container locally, their operating system must be able to run Docker (Linux, Mac, and Windows).

2. Docker on Linux requires kernel 3.10 or higher. No memory requirements are listed, but it can be assumed that the 4GB listed for Mac and Windows applies.

3. Docker on Mac requires macOS 10.14 or newer, and 4GB of RAM.

4. Docker on Windows requires WSL2, Windows 10, a 64-bit processor with Second Level Address Translation, 4 GB of RAM, and BIOS-level hardware virtualization must be enabled. *(Docker Inc., n.d.)*

If the user wants to connect to a remotely hosted Docker container, then the user only needs access to an application that supports the Remote Desktop Protocol. Microsoft Remote Desktop is

available for Windows and Mac, but alternative applications are available for other operating systems. For example; Remmina for Linux.

2.5 Design and Implementation Constraints

- The product must support Flutter and Dart. There are no other constraints have been identified
- To fully utilize emulator support, the system must be capable of nested virtualization

2.6 User Documentation

Developers might not be familiar with the use of Docker, the use of VS Code, the use of the Flutter/Dart executables, and GIT. The DevSecOps will provide minimal instructions via the user's guide.

2.7 Assumptions and Dependencies

- It is an assumption at this point in time that ADF can be deployed to and connected to AKS. Kubernetes is a container manager that can be used to scale containers on demand. This over works well as it keeps costs low and scales to handle demand. The ADF container can be host in Azure for world wide access via a simple RDP connection. The assumption is that the containers in azure can easily be connected to be end-users while scaling to handle current loads.
- ADF has many dependencies as the project itself just combines of the shelf technologies in such way as to provide the end-user with a functional development environment. Here is the list of dependencies:
  - GIT
  - VS Code
  - Android Emulator
  - Android SDK
  - Flutter Command Line Utilities
  - Dart SDK
  - AKS
  - Docker
  - .Net Runtime
  - .Net SDK

- Pulse Audio
- XRDP
- XFCE4

3. System Features

3.1 Conduct Git Operations

The user needs to be able to load code that they previously worked on, and commit code changes.

3.2 IDE Supports Project Type

The user's local environment supports specified project type as required for the class.

3.3 IDE Supports IntelliSense

Developers have a need for automatic code formatting. Flutter and Dart plugins in VS Code take care of this need.

3.4 Build App

The system needs to be able to compile and build the application. Flutter/Dart CLI executables provide this functionality.

3.5 Run Tests

Both developers and testers need to run test against the code. This can be done from the terminal inside VS Code using the Flutter/Dart executables in the container.

3.6 Remotely accessible

The environment can run in the cloud or locally, and in both cases the user can connect to the system using the remote desktop protocol.

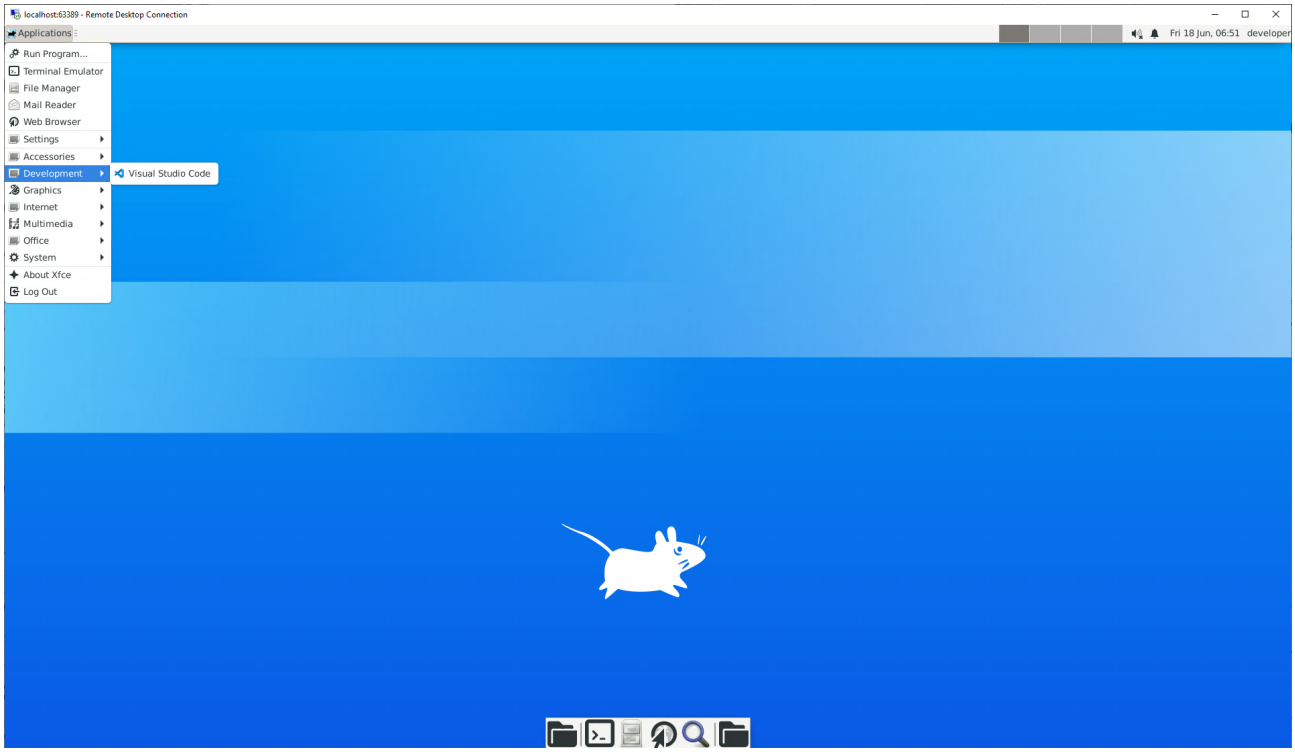
3.7 Code Analysis Support

Clean code if vital to maintainability and health of the application. The automated scan tool provided by SonarCloud will be used to conduct the wide analysis of code provided the actors. The system must provide an easy use command to execute the command to SonarCloud for analysis execution and return the results of the scan to the user.

4. External Interface Requirements

4.1 User Interfaces Overview

The user will connect to the system using their RDP client of choice. After connecting, the user sees a login screen, where they enter the default username and password “developer” and “password”. This brings them to the XFCE desktop. From here, they can interact with the system to start a Visual Studio Code. Other applications that are available from the desktop are a terminal emulator, and a web browser. VS Code also provides a terminal.



4.2 Hardware Interfaces

The intended endpoint to run this product is Azure Kubernetes (AKS). AKS is designed to scale on demand. With that being said the user interfaces with this capability via Remote Desktop Procedure.

Each endpoint in AKS needs to be able to route port 3389.

4.3 Software Interfaces

The Docker image for the ADF contains a Linux installation including all requires software. The following software packages are installed:

Name	**Purpose**
.Net runtime	Runtime
Android Command Line Utilities	Needed for Dart/Flutter
Android Emulator	Needed for Dart/Flutter
Android SDK	SDK
Dart SDK	SDK
DotNet SDK	SDK
Flutter SDK	SDK
GIT	CLI utility for interaction with GIT repository
Google Chrome	Web browser
NodeJS	Server-side Javascript
OpenJDK	Java development kit and runtime
PulseAudio	Sound support
QEMU	Machine emulator, required for Android emulator
SonarScanner	Local SonarCloud scanning
Visual Studio Code	IDE
X11	X Window System
XFCE4	Lightweight desktop environment
XRDP	Remote desktop server

5. System Features/Modules

5.1 Conduct Git Operations

5.1.1 Description and Priority

The user needs to be able to load code that they previously worked on, and commit code changes. Code is stored on GitHub. The user will use the git command line tool to interact with the repository. Commands include git clone, git pull, git push, git branch, git add, git commit, and so on. This is a high priority, as it is essential to use any of the other features.

5.1.2 Stimulus/Response Sequences

Stimulus: User uses terminal to issue a git command.
Response:Git command is executed.

5.1.3 Functional Requirements

REQ-1.1: Upon issuing a git command, the system shall execute the git command.

5.2 IDE Supports Project Type

5.2.1 Description and Priority

The user's local environment supports specified project type as required for the class. This requires Flutter and Dart to be installed, but also supporting applications for Flutter/Dart, such as an Android emulator. This is of high priority.

5.2.2 Stimulus/Response Sequences

Stimulus: In a terminal the user issues a flutter command.
Response:The flutter command is executed.

Stimulus: In a terminal the user issues a command to run the Flutter app.
Response:The app runs in a browser or Android emulator.

5.2.3 Functional Requirements

REQ-2.1: Upon entering a valid flutter command, the command shall be executed.
REQ-2.2: Upon entering a command to run a Flutter app, the app shall be executed in a browser or Android emulator.

5.3 IDE Support IntelliSense for Specified Projects

5.3.1 Description and Priority

Developers have a need for automatic code formatting. Flutter and Dart plugins in VS Code take care of this need. This is of medium priority, as the system still has usefulness without this feature.

5.3.2 Stimulus/Response Sequences

Stimulus: In VS Code, the user right clicks, and choose Format Document.
Response: The current file is formatted.

Stimulus: The user uses a shortcut to format the document, such as Shift-Alt-F.
Response: The current file is formatted.

Stimulus: The user chooses Format Document from the VS Code Command Palette.
Response: The current file is formatted.

5.3.3 Functional Requirements

REQ-3.1: Upon an interaction with the VS Code user interface that start the document formatter, the current file shall be formatted according to Flutter/Dart standards.

5.4 Build App

5.4.1 Description and Priority

The system needs to be able to compile and build the application. Flutter/Dart CLI executables provide this functionality. This is of high priority.

5.4.2 Stimulus/Response Sequences

Stimulus: The user issues a build command in a terminal.
Response: The app builds.

5.4.3 Functional Requirements

REQ-4.1: Upon entering the appropriate build command in a terminal, the app shall be built.

5.5 Run Tests

5.5.1 Description and Priority

Both developers and testers need to run test against the code. This can be done from the terminal inside VS Code using the Flutter/Dart executables in the container. This is of medium priority.

5.5.2 Stimulus/Response Sequences

Stimulus: The user issues a flutter command to run unit tests.
Response: The unit tests are executed, and the result is displayed.

5.5.3 Functional Requirements

REQ-5.1: Upon entering the appropriate flutter command in a terminal to execute the unit tests in the project's test directory, the tests shall be executed, and the results shows to the user.

5.6 Remotely accessible

5.7.1 Description and Priority

The system can run in the cloud, or locally. In both cases, the user connects to the system using the remote desktop protocol.

5.7.2 Stimulus/Response Sequences

Stimulus: The user connects with an RDP client to localhost.
Response: The user sees the user interface.

Stimulus: The user connects with an RDP client to the correct URL.
Response: The user sees the user interface.

5.7.3 Functional Requirements

REQ-7.1: Upon connecting with an RDP client to either localhost when ran locally, or a remote URL in the case of a cloud-hosted container, the user shall see the user interface for the ADF.

5.7 Code Analysis

5.8.1 Description and Priority

Clean code if vital to maintainability and health of the application. The automated scan tool provided by SonarCloud will be used to conduct the wide analysis of code provided the actors. The system must provide an easy use command to execute the command to SonarCloud for analysis execution and return the results of the scan to the user. This is of medium priority.

5.8.2 Stimulus/Response Sequences

Stimulus: The user issues a sonar-scanner command in a terminal.
Response: The code is analyzed by SonarCloud.

5.8.3 Functional Requirements

REQ-8.1: Upon entering a valid sonar-scanner command in a terminal, the code is analyzed by SonarCloud.

6. Nonfunctional Requirements

6.1 Performance

NF-1.1: There system responds quickly to user input. There is minimal lag.

6.2 Security

NF-2.1: No usernames or passwords are hardcoded or stored in the container.