Harris Healthcare Workload Application Software Configuration Management Plan

COSC 471
Computer Science Department
Okanagan College

Version 1.9

Taylor Adam
Nelson Murray
Grayson King
Jackson Bates
Christian Slater
George Cairney
Dakota Logan
Brandon Chesley
Daniel Claggett

April 2020

Revision History

Revision	Date	Brief Summary of Changes	Author
Version 1.4	2020/01/08	Update formatting to match other documents	N. Murray
Version 1.5	2020/02/11	Updated continuous integration section	T. Adam
Version 1.6	2020/02/25	Added continuous deployment section	T. Adam
Version 1.7	2020/03/10	Added Create web app bot service on Azure section, Added image to Deploy app from Visual Studio 2019 section	T. Adam
Version 1.8	2020/03/26	Updated Version	T. Adam
Version 1.9	2020/04/06	Added The Mock Api Configuration	T. Adam

Contents

1 Introduction

This Software Configuration Management plan for the Harris Healthcare Workload Application outlines the configuration for our source code, deployment, version control, and continuous integration.

2 Source Code

2.1 Outline

To be determined

2.2 Standards

2.2.1 Variables

Fields (Non-private, private) Camel case, Start with a capital letter.
Fields (Instance, private) Camel case, start with _ followed by a lowercase.
Local Variables Camel case, start with a lowercase letter.

2.2.2 Functions

Function(Non-private, Private) - Camel case, Start with a capital letter, XML comments above

Asynchronous Function(Non-private, Private) - Camel case, Start with a capital letter, ends with "Async" , XML comments above

2.2.3 Filenames

Service Class - Camel case, start with a capital letter, end with Service
Component - Camel case, start with a capital letter, end with Component
Interface - Camel case, start with a capital I
Enum - Camel case, start with a capital letter, end with Enum
Helper Class - Camel case, start with a capital letter, end with Helper, must be static

2.3 Headers

Headers are to be included in all source code files, and are to be updated whenever anyone updates them. See outline below:

```
/* File = example.cs
   Author = Taylor Adam
   Date = 2020/01/08
   License = MIT
   Modification History
   Version          Author          Date             Desc
```

```
        v 1.0          Taylor Adam    2020/01/08    Added Headers  */
```

2.4 Comments

Detailed XML comments can be found for every Method and Constructor, including parameters and, if applicable, what is returned. An example comment can be seen below:

```
/// <summary>
/// Method invoked when the component is ready to start, having received its
/// Gets the forms from the form services and assigns them to FormsList
///</summary>
/// <returns>A Task</returns>
```

2.5 Design

To be determined

3 Deployment

* Note - You will need to do this for HarrisPharamacy.API as well if deploying aplongside the Mock API

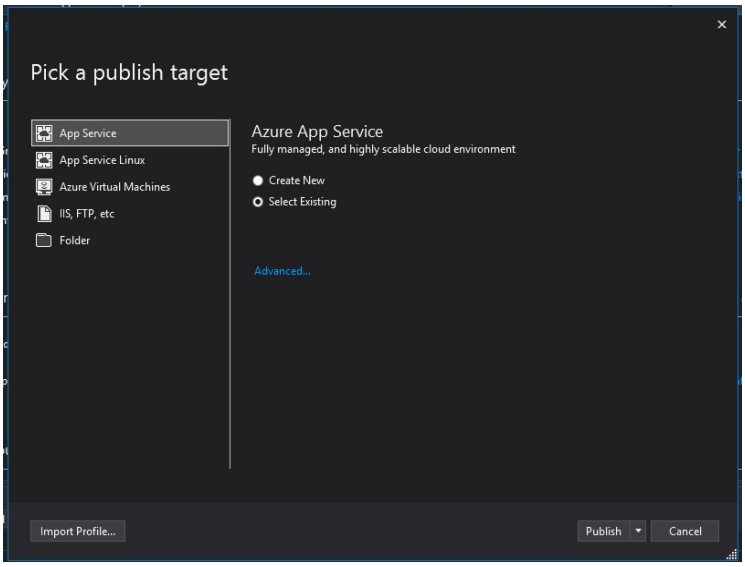
3.1 Azure Deployment

3.1.1 Create a new web service on Azure:

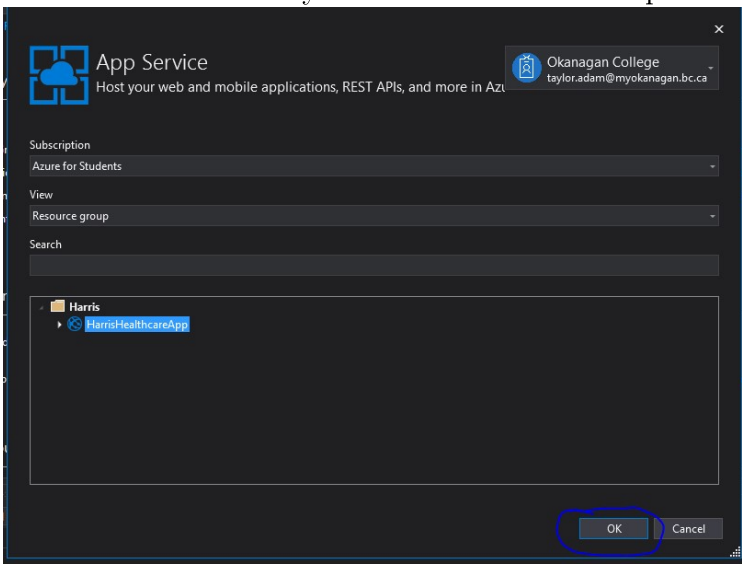
1. Navigate to https://portal.azure.com/# create/Microsoft.WebSite
2. Pick your subscription, for instance mine is Azure for students.
3. Pick your resource group or create a new one
4. Create a name
5. Set publish option to Code
6. Set run time stack options to .Net Core 3.0
7. Set Operating system to Windows
8. Set Region to preferred Region (I am using Canada Central)
9. Create new app service plan (Might have already done that for you)
10. For SKU and size select change size and pick Free F1
11. Review and create

3.1.2 Deploy app from Visual Studio 2019:

1. Open app in Visual Studio 2019 (Any 2019 edition should work)
2. Click the build drop down-¿ publish
3. Choose App service; select existing



4. Make sure you are signed into the same Microsoft account you were using to create the Azure web service.
5. Find the web service you made in the first step of this guide and click OK.



3.1.3 Create web app bot service on Azure:

1. Navigate to https://portal.azure.com/#create/Microsoft.BotServiceSdkGalleryPackage
2. Pick your subscription
3. Pick your resource group or create a new one
4. Pick a pricing tier, F0 for free
5. Give the app a name
6. Choose a bot template, anyone will work
7. Set Region to preferred Region (I am using Canada Central)
8. Create new app service plan (Might have already done that for you)
9. Click create

3.2 Local Deployment With IIS

Please see https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/?view=aspnetcore-3.1 for a detailed guide on hosting ASP.NET Core on Windows with IIS.

4 Version Control

We use git for version control. We make branches for different features and name them accordingly. We commit to staging(repository on GitHub) first to test the web app and make sure it is running as intended, then we merge those changes onto origin master(repository on bit-bucket) to be deployed to the production server.

5 Continuous Integration

We will be using a GitHub pipeline pipe to azure web apps deploy as our CI tool. To achieve this we have created a private GitHub repository (https://github.com/TaylorAdamCA/HarrisPharamacyTestEnviroment) that is used for our staging repository. Every time we commit to this repository, the pipe is run automatically (including all tests) and the web app is deployed to https://harristesting.azurewebsites.net .

6 Continuous Deploymenmt

We will be using a Github pipeline pipe to azure web apps deploy as our CD tool. To achieve this we have created a branch on the Emerald-Health-Information/HarrisPharmacyApp repository called dev, that will run this pipeline to automatically deploy our development branch to https://harrisapp-dev.azurewebsites.net/. We also are going to set up a CD pipline for the master branch, that we will use for realeses.