

DevSecOps
Programmer's Guide

Jeroen Soeurt
Michelle Monfort
Robert Wilson

University of Maryland Global Campus
SWEN 670 Software Engineering Project
Professor Dr. Mir Assadullah
Summer 2021

Table of Contents

Table of Figures:	3
Introduction	5
GitHub	7
Introduction	7
Installing Git	7
Windows	7
Linux	10
MacOS	10
Basic Git usage	10
GitHub Initial Configuration	10
Advanced Development Factory	11
Introduction	11
Installing Docker	11
Windows	11
Linux	12
MacOS	12
Clone the ADF repository	12
From a terminal, clone the ADF repository using.....	12
Visual Studio Code:	14
Build the Docker image	15
Running the ADF locally	15
Connecting to the running container	15
The ADF Dockerfile	20
The ADF welcome app.....	21
Introduction	21
Cloning the repository.....	21
Running the app locally	21
Building the app	21
Structure	22
Programming for CaPPMS.....	22
Programming CaPPMS with ADF	22

Programming CaPPMS with Windows and Visual Studios	23
Getting Visual Studio	23
Developing for CaPPMS.....	30
File Structure	31
Package Manager	33
Using Attributes	34
Page Flow	37

Table of Figures:

Figure 1: GIT Setup Screen 1 - License	8
Figure 2: GIT Setup Screen 2 - Common Options	9
Figure 3: GIT Setup Screen 3 - Use Unix style line endings	10
Figure 4: Docker - Images.....	12
Figure 5: Docker - GitHub Repository Sync	13
Figure 6: Docker - Edit startup.sh.....	13
Figure 7: Docker - Notepad++ EOL Conversion	14
Figure 8: Code EOL location on status bar	14
Figure 9: Code EOL Selection.....	14
Figure 10: Docker - Microsoft Remote Desktop Connection Screen.....	15
Figure 11: Docker - Remmina Remote Desktop Preference Screen Part 1	16
Figure 12: Docker - Remmina Remote Desktop Preference Screen Part 2	17
Figure 13: ADF - Login Screen.....	18
Figure 14: ADF - Welcome Screen via Remmina on Linux.....	19
Figure 15: ADF - Welcome Screen via Remote Desktop Connection on Windows	19
Figure 16: ADF Welcome Screen	21
Figure 17: ADF Terminal - Start VS Code	22
Figure 18: VS Code - Load C# Extension	23
Figure 19: VS Setup - Select Asp.Net	24
Figure 20: VS Setup - Select .Net cross-platform development	24
Figure 21: VS Setup - Customize Visual Experience	25
Figure 22: Git Clone Command	25
Figure 23: GitHub Code - Clone URL Location	26
Figure 24: Load CaPPMS from Command Window	26
Figure 25: Application Selector	27
Figure 26: Git Changes Prompt	27
Figure 27: Git User Information Prompt	28
Figure 28: VS Start Debug on IIS Express.....	28
Figure 29: SSL Configuration Prompt	28
Figure 30: SSL Certificate Install Prompt	29
Figure 31: Debug CaPPMS.....	29
Figure 32: Application Order	31

Figure 33: File Structure	32
Figure 34: VS Tools - Nuget Manager	33
Figure 35: VS Nuget Manager	34
Figure 36: Annotations.....	34
Figure 37: SpanIcon Attribute Example.....	35
Figure 38: SpanIcon Example - GitHub as Button.....	36
Figure 39: SpanIcon Example - GitHub as Anchor	36
Figure 40: SpanIcon - Implementation	37
Figure 41: Component - Cascading Parameters	38
Figure 42: File Structure – ProjectTable.razor Call Out	39
Figure 43: Component - Directives.....	39
Figure 44: Table Model	40
Figure 45: C# code wrapped in HTML	41

Introduction

Welcome to Capstone! You've made it this far and are now wondering how to tie it all together. This document cannot give you the answer to that but can maybe help by providing the steps the previous class did to get there.

There are inter-related documentation guides that might be worth going back to. None of these guides are written to be a lock step execution matrix. Not everything in the documentation will have to be done and some items you can completely disregard depending on the conditions given to you.

It is important to research the conditions given before you start coding anything! Technology that maybe required only works with some technology and not others. An example of this is the analysis tool known as SonarCloud. You will find steps in the guides provided for you. It is worth noting that SonarCloud does not work with all programming languages and therefore might not be applicable to your set of given conditions.

Now, guides? Throughout our term we produced several guides. Each of these guides have flaws but are meant to be helpful. Truth is at the time of writing, there is no way to determine how this, or any other guide will impact the reader. With that in mind, this document is written from the perspective of what the picture looks like now. Here is a quick rundown:

- Programmer's Guide (this document) – Meant to get your personal development kick started in developing for ADF (Advanced Development Factory) and CaPPMS (Capstone Management System).
- User's Guide – How to use ADF and CaPPMS
- Runbook – How to set up Cloud technologies

As you look through the documentation there are some overlapping themes. Knowing your way around Git and GitHub is vital to your success. There are multiple ways to navigate Git but for the most part command line was given as it translates well across platforms. Good luck!

GitHub

Introduction

The version control system used in this course is Git. The use of Git version control system was developed about twenty years ago. The benefits version control systems are that they allow users to manage and keep track of source code and history. While you could run your own git server to host repositories, it is much easier to use a free hosted service. This project implemented the git version control system GitHub. GitHub a cloud-based hosting service that allows users to manage Git repositories. The assumption is that future programmers will implement the same tools. In the event that programmers decide they would like to use other tools alternatives to GitHub could be Atlassian BitBucket or Microsoft Azure DevOps. The benefit of a hosted service over running your own is that these services offer more than just version control. Agile project management tools and pipelines for example.

Installing Git

Windows

Navigate to <https://git-scm.com/> and download the latest installer.

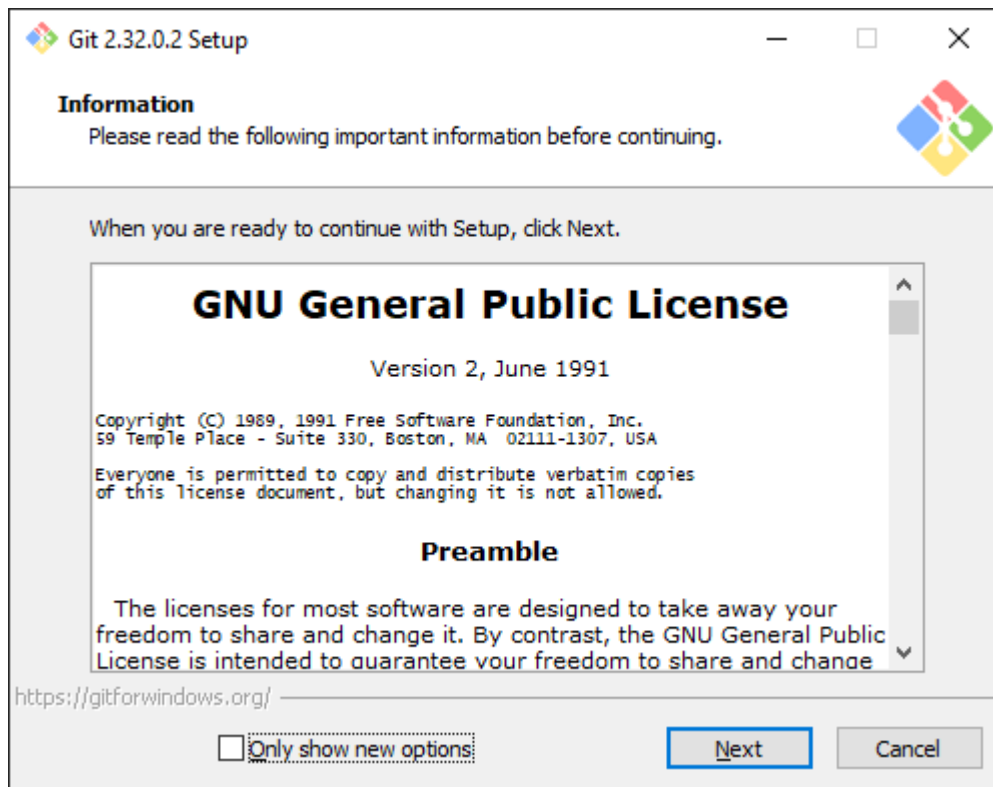


Figure 1: GIT Setup Screen 1 - License

Accept the GPL by clicking next. The default options on the next screen can be left as-is.

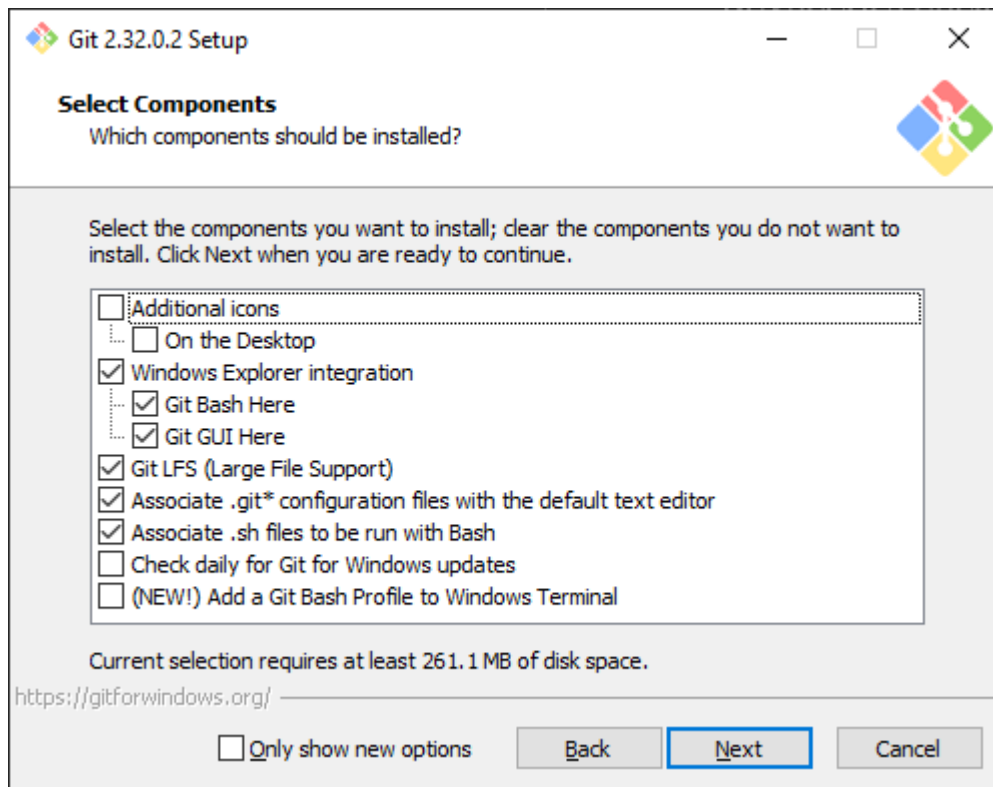


Figure 2: GIT Setup Screen 2 - Common Options

Click next several times until you get to this screen:

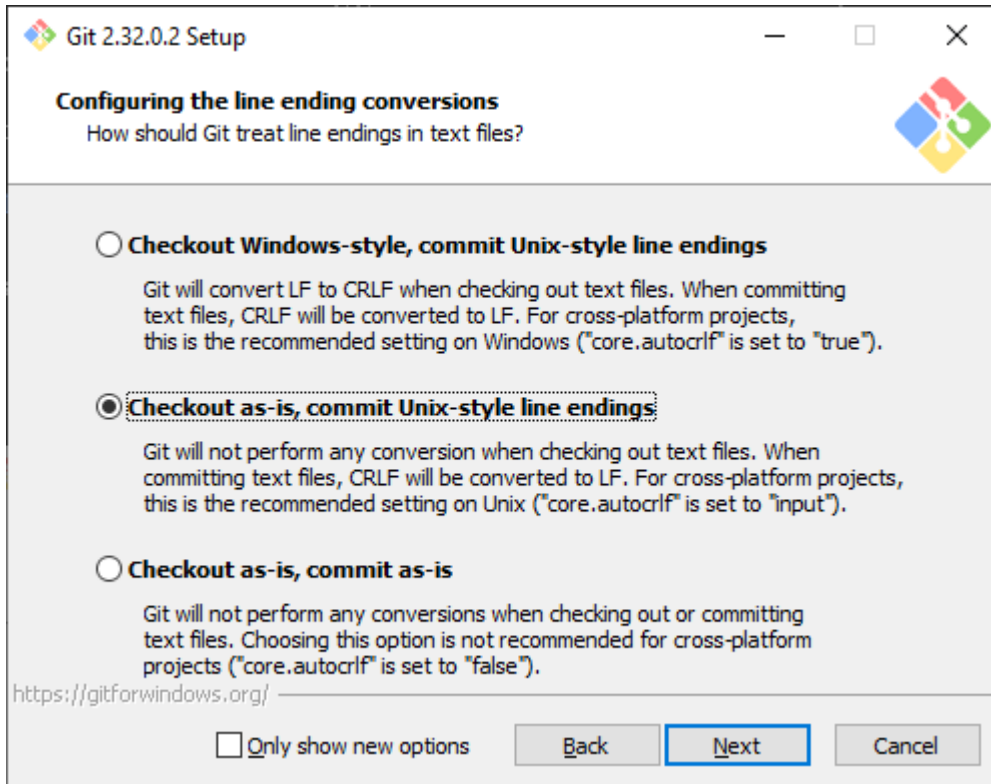


Figure 3: GIT Setup Screen 3 - Use Unix style line endings

Here it is important to select Checkout as-is. Otherwise, you can run into problems when cloning the Linux/Unix/Mac based projects on a Windows device. All other default options on the next screens are ok as-is. Click install to complete the installation.

Linux

Git is included in most distributions. For Debian based (Ubuntu, Mint) use apt-get:

- `sudo apt-get install git`

For RPM based distros (RedHat, Centos, Fedora), use yum:

- `sudo yum install git`

MacOS

Git is included with Xcode, so you could install XCode from the app store. If you don't want to install Xcode, you can also install Git using Homebrew:

- `brew install git`

Basic Git usage

Please refer to the user guide for examples and documentation on basic end-user Git usage.

GitHub Initial Configuration

Initial GitHub configuration was provided by course mentors, in which case you submit your already established GitHub account and are assigned teams. New GitHub Account can be created at <https://github.com> for users that do not have a GitHub account

Advanced Development Factory

Introduction

The Advanced Development Factory, or ADF, is a fully containerized development environment. It contains a full Linux desktop system with several browsers, Visual Studio Code, Android development tools, Flutter, Dart, and the Android emulator. DevSecOps has also installed several other tools that might help future classes, such as Dotnet.

Installing Docker

The first step to get the ADF running on your local machine is to install Docker. Docker is a containerization platform that runs on Windows, Mac, and Linux.

Windows

Install Docker Desktop from <https://hub.docker.com/editions/community/docker-ce-desktop-windows>.

Most Docker containers require a Linux kernel. Windows facilitates this through Windows Subsystem for Linux.

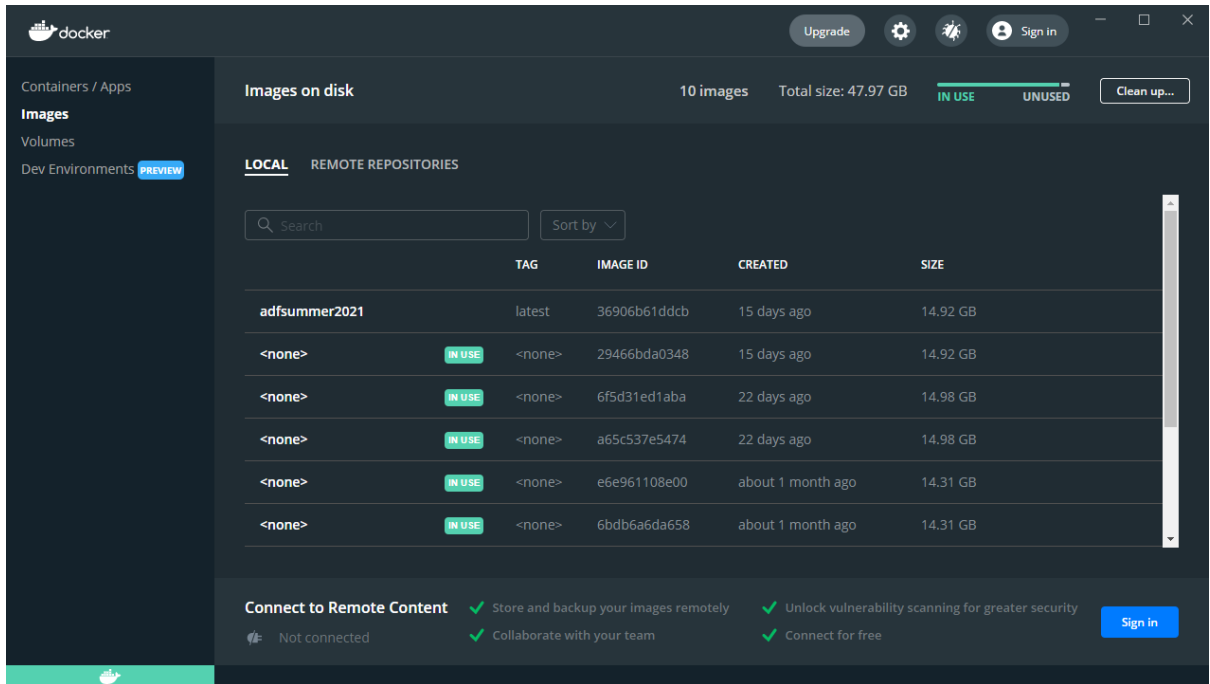


Figure 4: Docker - Images

Linux

Follow the steps for your distribution at <https://docs.docker.com/engine/install/>.

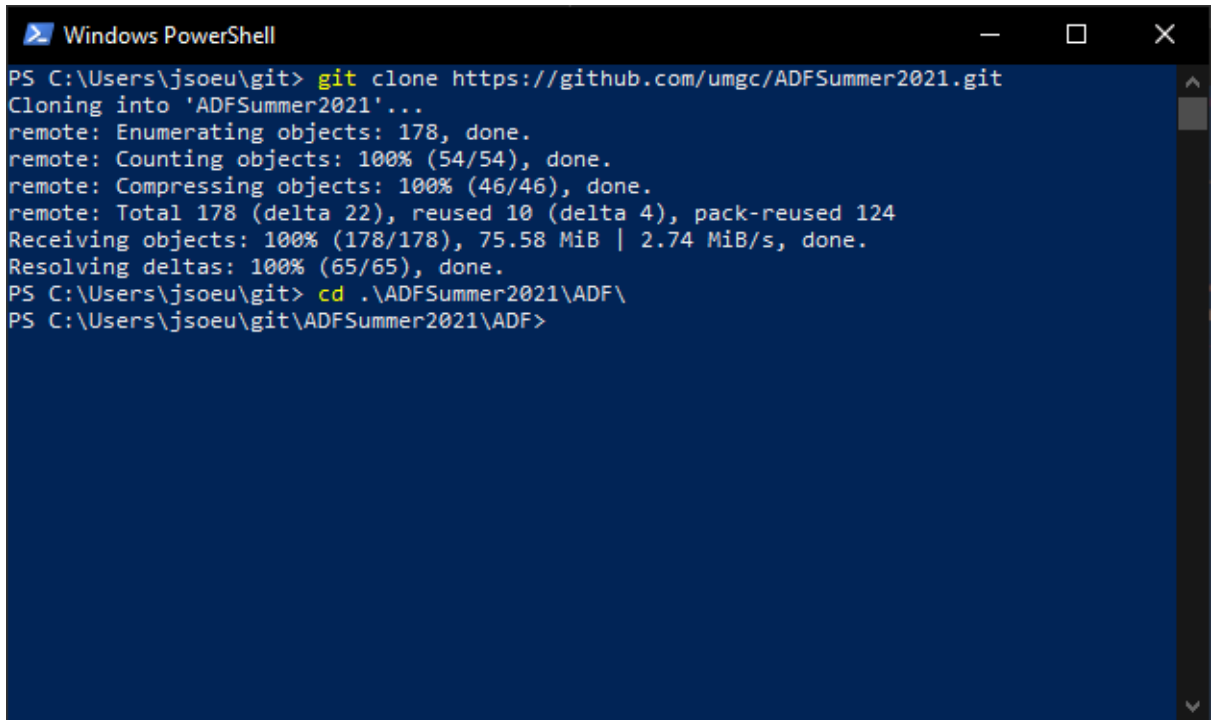
MacOS

Install Docker Desktop from <https://hub.docker.com/editions/community/docker-ce-desktop-mac>.

Clone the ADF repository

From a terminal, clone the ADF repository using

- `git clone https://github.com/umgc/ADFSummer2021.git`
- `cd .\ADFSummer2021\ADF\`

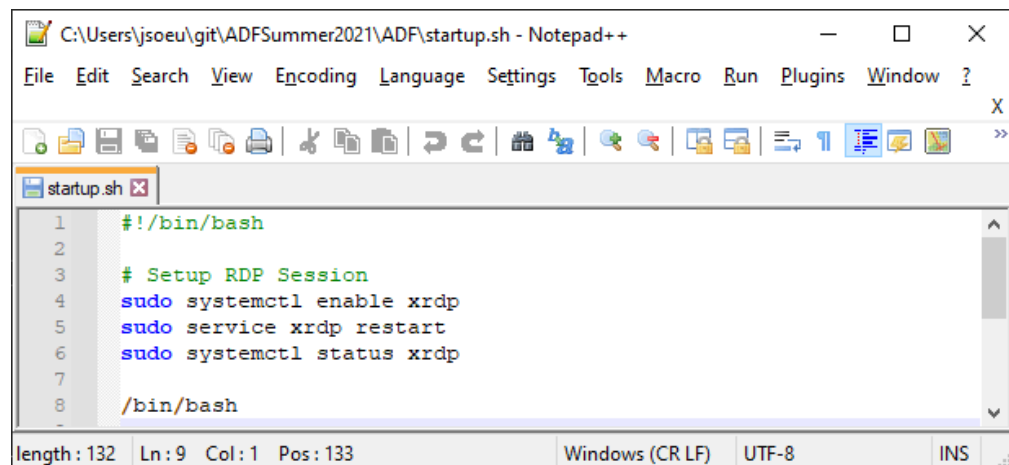


```
PS C:\Users\jsoeu\git> git clone https://github.com/umgc/ADFSummer2021.git
Cloning into 'ADFSummer2021'...
remote: Enumerating objects: 178, done.
remote: Counting objects: 100% (54/54), done.
remote: Compressing objects: 100% (46/46), done.
remote: Total 178 (delta 22), reused 10 (delta 4), pack-reused 124
Receiving objects: 100% (178/178), 75.58 MiB | 2.74 MiB/s, done.
Resolving deltas: 100% (65/65), done.
PS C:\Users\jsoeu\git> cd .\ADFSummer2021\ADF\
PS C:\Users\jsoeu\git\ADFSummer2021\ADF>
```

Figure 5: Docker - GitHub Repository Sync

Important: If you are on Windows, and installed Git with Checkout Windows Style, the line-endings of `startup.sh` will be set to CRLF. This means that when the file is copied into the Linux container at the end of the build, it will cause an error. To fix this, download Notepad++ from <https://notepad-plus-plus.org/downloads/>. Open up the file. At the bottom it will show you Windows (CR LF) in the status bar:

- Notepad++



```
C:\Users\jsoeu\git\ADFSummer2021\ADF\startup.sh - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
startup.sh
1  #!/bin/bash
2
3  # Setup RDP Session
4  sudo systemctl enable xrdp
5  sudo service xrdp restart
6  sudo systemctl status xrdp
7
8  /bin/bash
length: 132 Ln: 9 Col: 1 Pos: 133 Windows (CR LF) UTF-8 INS
```

Figure 6: Docker - Edit startup.sh

Choose `Edit > EOL Conversion > Unix (LF)`, and hit save.

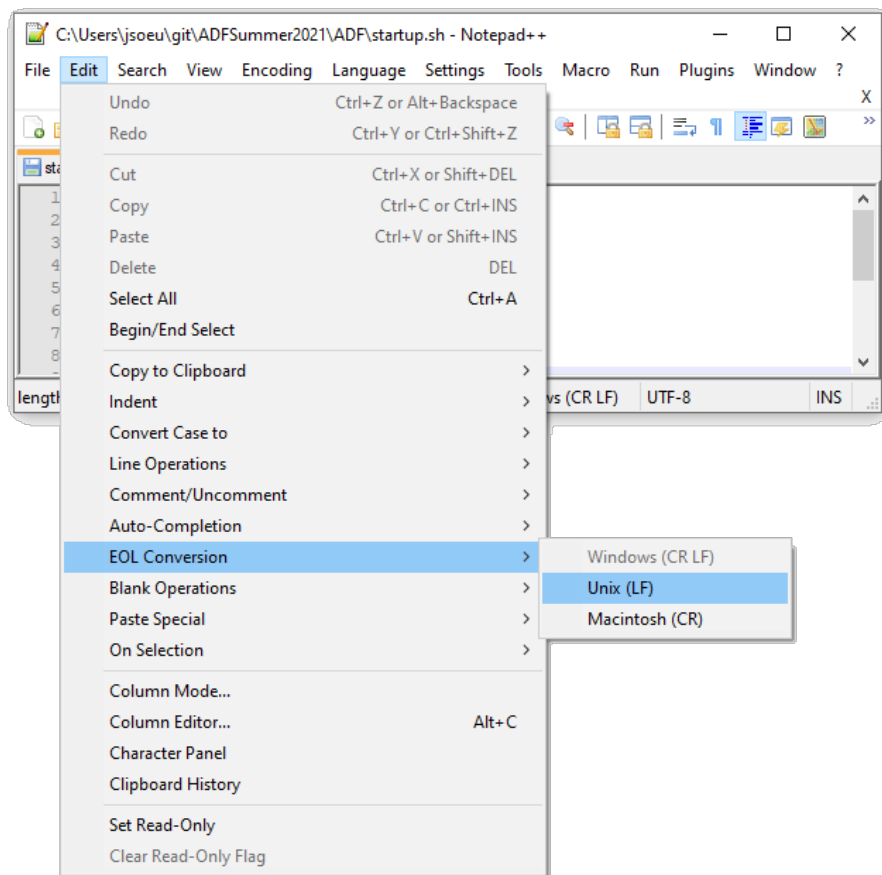


Figure 7: Docker - Notepad++ EOL Conversion

Visual Studio Code:

Click the CRLF/LF from the status bar located in the bottom right of the screen:



Figure 8: Code EOL location on status bar

Select “LF”

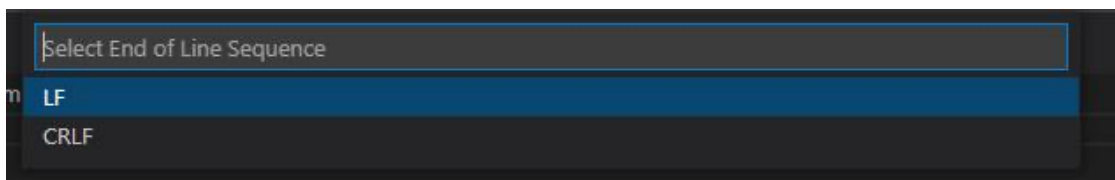


Figure 9: Code EOL Selection

Build the Docker image

From the command line, run the following command:

- `docker build --pull --rm -f "ADF/dockerfile" -t adfsummer2021:latest "ADF"`

This process will take a while depending on your internet connection.

Running the ADF locally

You can now run the image using:

- `docker run -dit -p 3389:3389 --rm --privileged adfsummer2021:latest`

Note: on Windows port 3389 might be occupied by the Windows Remote Desktop server. In that case, run it with 63389:3389, and in the next step connect to localhost:63389.

Connecting to the running container

Use your favorite RDP client to connect to localhost:3389 (or 63389 on Windows).

On Windows, use Remote Desktop:

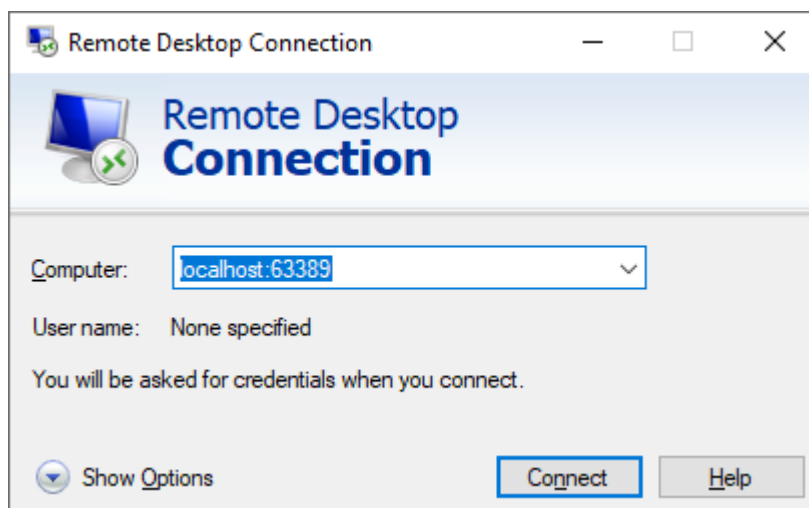


Figure 10: Docker - Microsoft Remote Desktop Connection Screen

On Linux, you can
install Remmina.

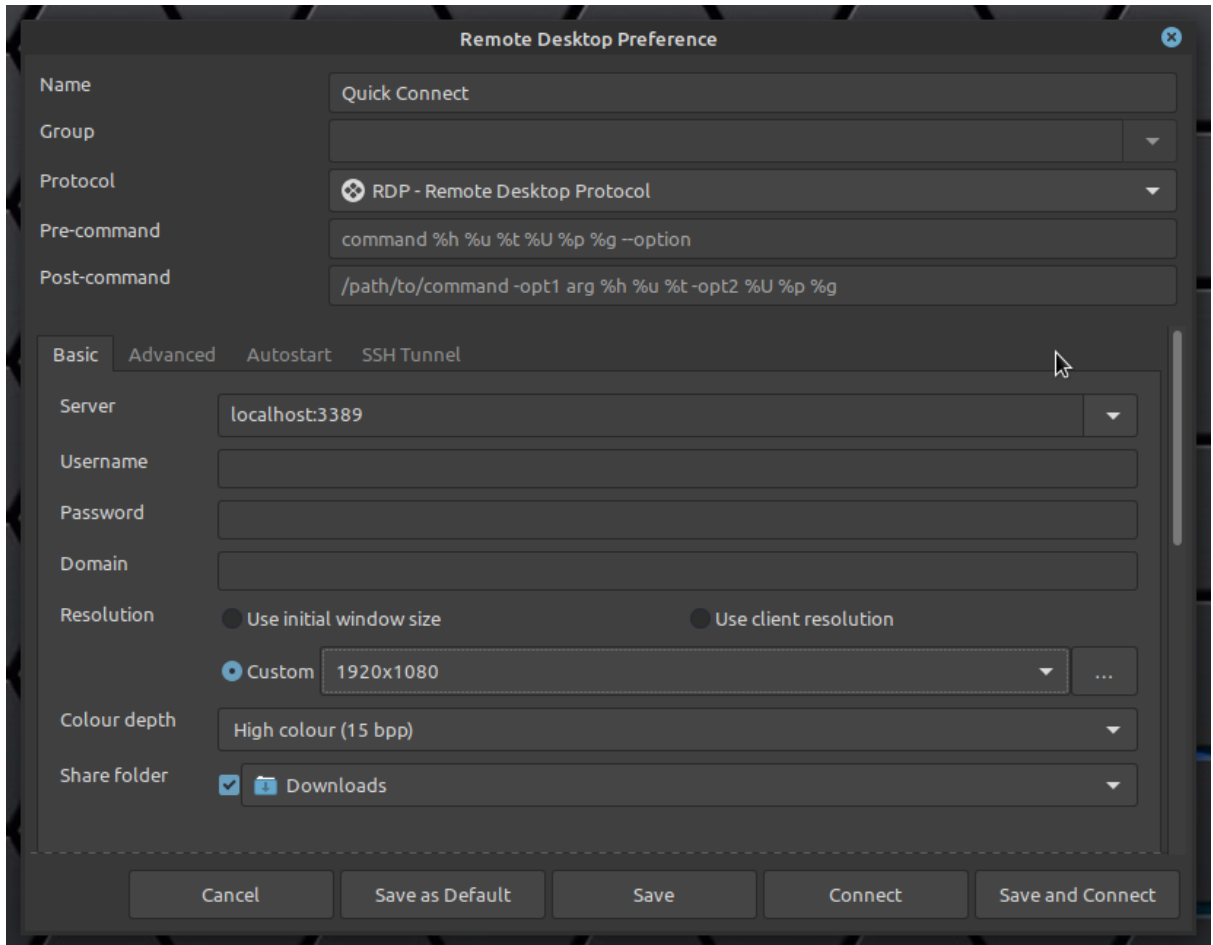


Figure 11: Docker - Remmina Remote Desktop Preference Screen Part 1

Make sure to enable sound:

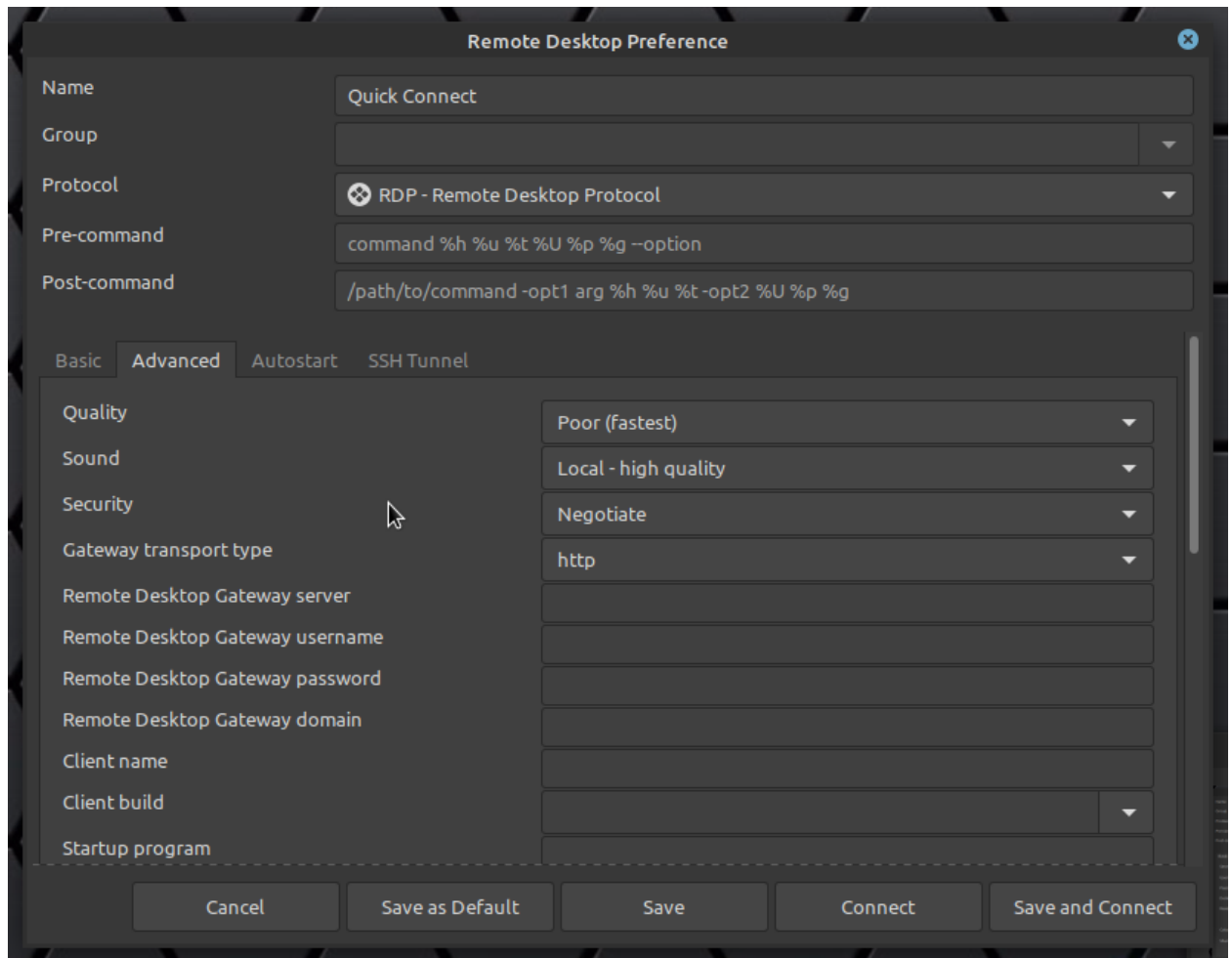


Figure 12: Docker - Remmina Remote Desktop Preference Screen Part 2

Next connect and you'll see a login screen.

- Username: developer
- Password: password

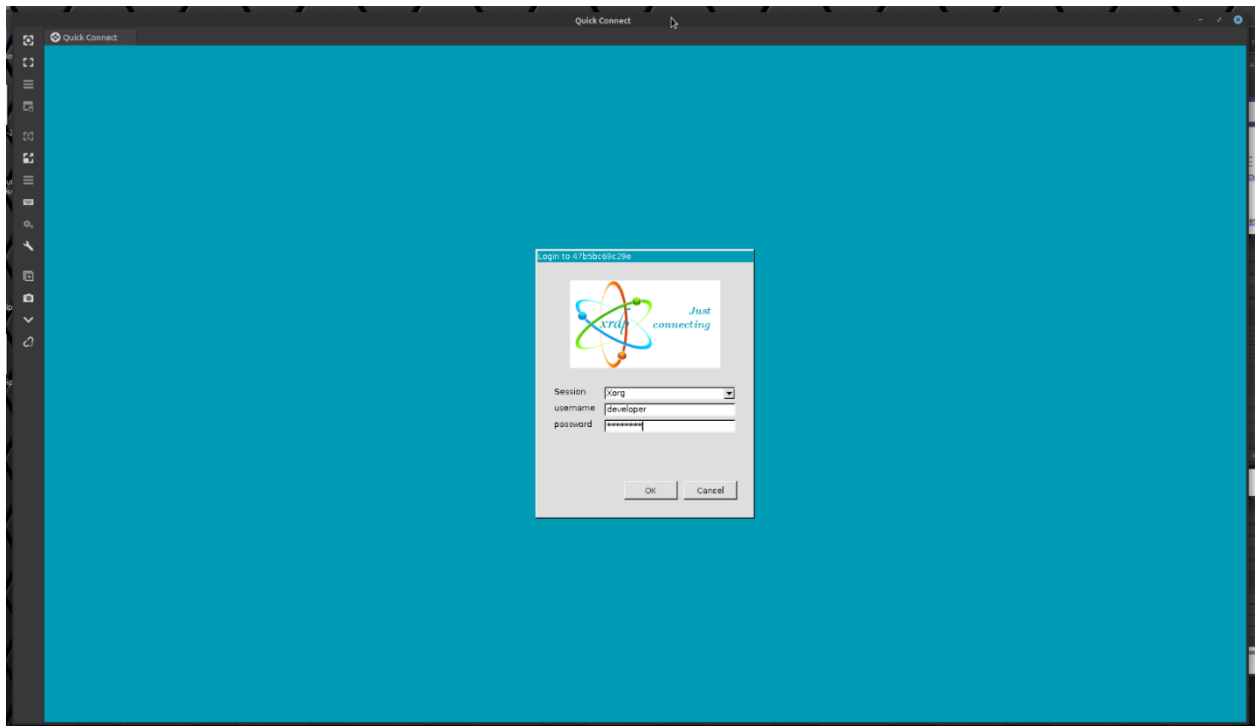


Figure 13: ADF - Login Screen

After logging in you'll be greeted by a welcome screen:
Linux:

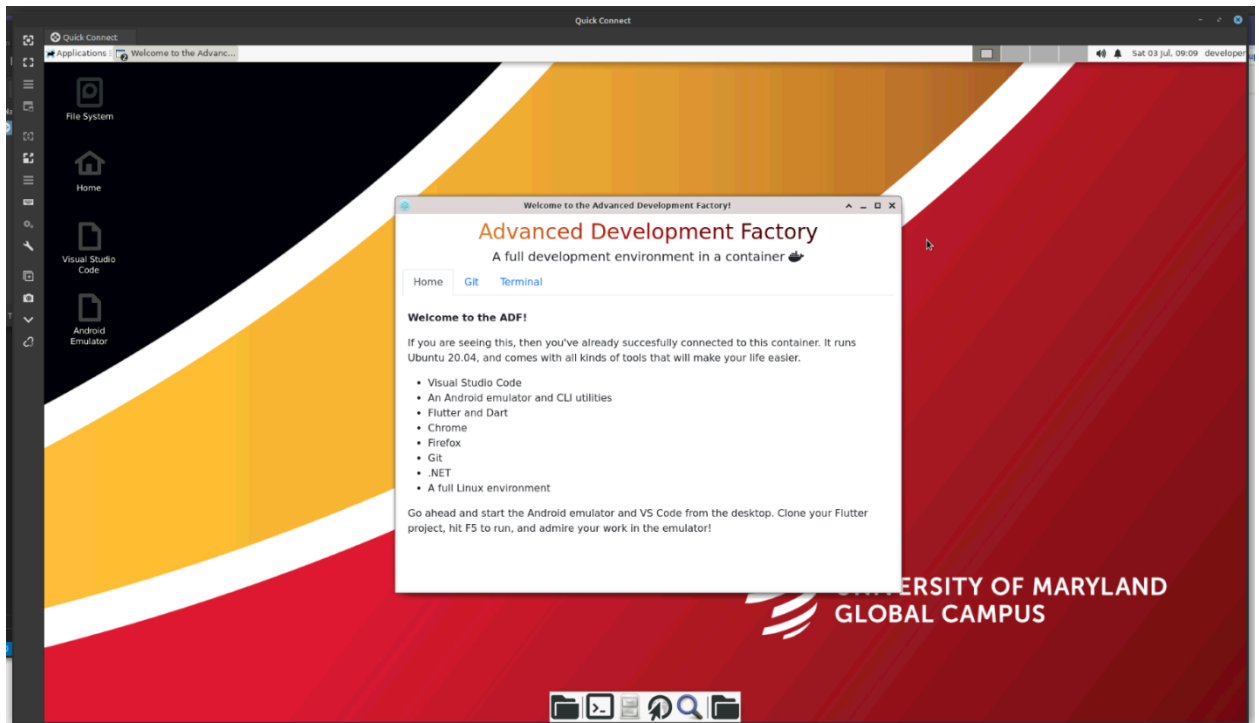


Figure 14: ADF - Welcome Screen via Remmina on Linux

Windows:



Figure 15: ADF - Welcome Screen via Remote Desktop Connection on Windows

The ADF Dockerfile

The ADF builds based on a file called a Dockerfile. This file contains all the steps needed, and can be found in /ADF/.

The file start from a base Linux image (Ubuntu LTS), installs software, adds users, and configures settings. The comments in the file will explain the individual sections.

Editing the [dockerfile](#) can be done with any text editor but it is recommended to use NotePad++ or VS Code as mentioned above due to the end of line control marker differences between platforms.

The ADF welcome app

Introduction

The ADF welcome app shows up when the user first logs in. It is written in HTML/JavaScript and uses Bootstrap. This is then converted into an executable using Electron.

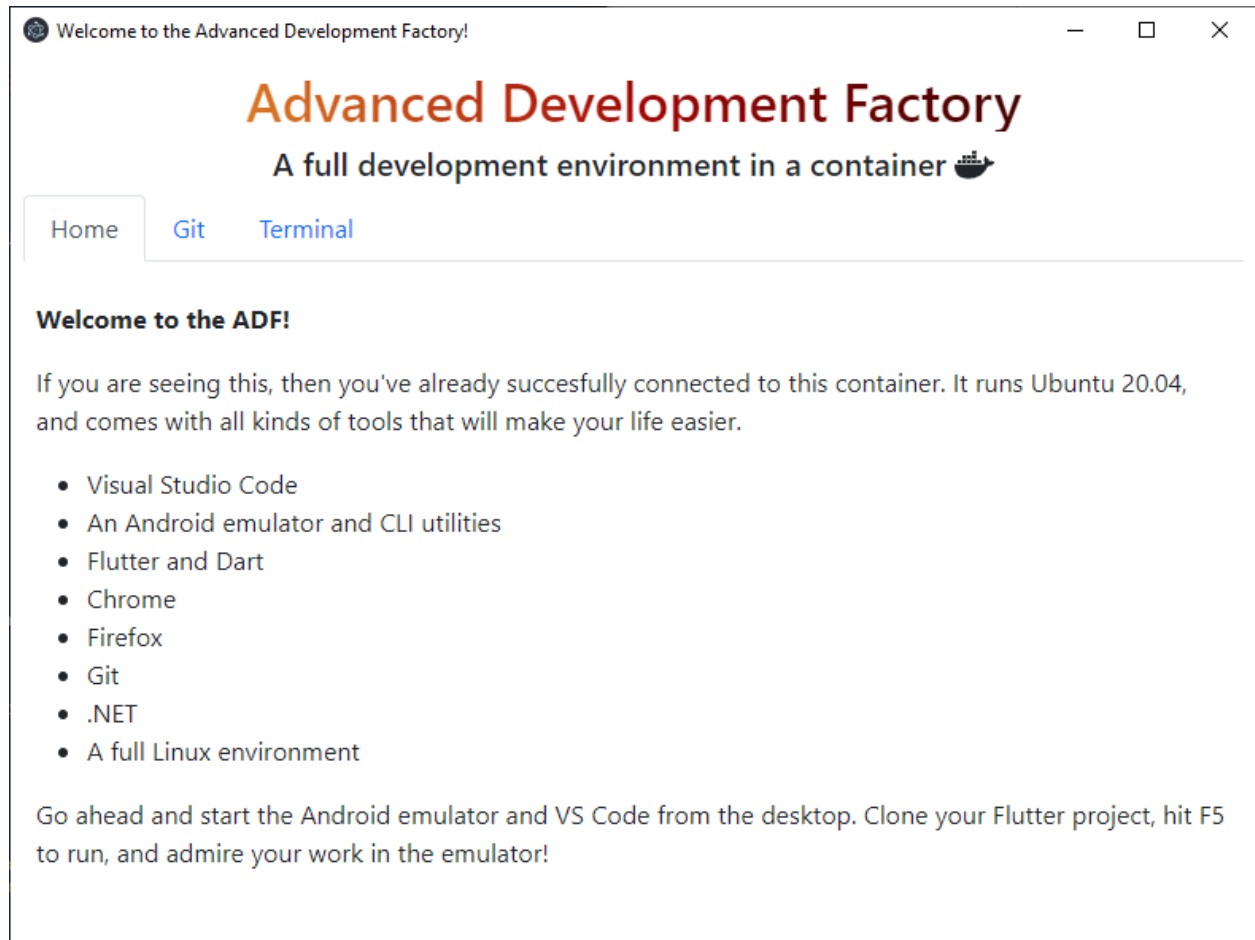


Figure 16: ADF Welcome Screen

Cloning the repository

The repository can be found on Github at <https://github.com/umgc/ADFWelcome>.

Running the app locally

After cloning, run `npm install` to install all dependencies.

Now you can run `npm start` to start the app locally.

Building the app

The app uses Electron Forge to build. Run `npm run make` to create the executable. The app builds for the environment it is on. So if you are on Linux, it will build for Linux. You can build the ADF app in the ADF.

The resulting executable can be found in the `/out` directory.

Structure

- All JavaScript is in `main.js`, and well documented.
- All content is in `index.html`
- `/node_modules/` contains all dependencies. NPM install will populate this, and there is no need to manually modify anything in here.
- `Package.json` contains the package info for NPM.

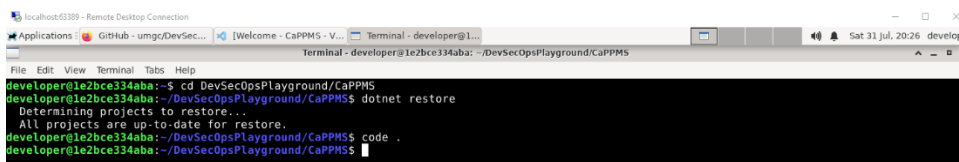
Programming for CaPPMS

CaPPMS can be developed using cross-platform technologies. The easiest way to get started programming CaPPMS is with ADF, mentioned in the next section.

Programming CaPPMS with ADF

Follow the previous mentioned step to get ADF operational on your host machine.

- Open the terminal and clone the Repository currently located at:
<https://github.com/umgc/DevSecOpsPlayground.git>
- `cd ./DevSecOpsPlayground/CaPPMS ->`
- `dotnet restore ->`
- `code . ->`



```
localhost:6389 - Remote Desktop Connection
Applications  Github - umgc/DevSec...  [Welcome - CaPPMS - V...  Terminal - developer@1...
terminal - developer@1e2bce334aba: ~/DevSecOpsPlayground/CaPPMS

File Edit View Terminal Tabs Help
developer@1e2bce334aba:~$ cd DevSecOpsPlayground/CaPPMS
developer@1e2bce334aba:~/DevSecOpsPlayground/CaPPMS$ dotnet restore
Determining projects to restore...
All projects are up-to-date for restore.
developer@1e2bce334aba:~/DevSecOpsPlayground/CaPPMS$ code .
developer@1e2bce334aba:~/DevSecOpsPlayground/CaPPMS$
```

Figure 17: ADF Terminal - Start VS Code

If prompted to install the C# extension, click install.

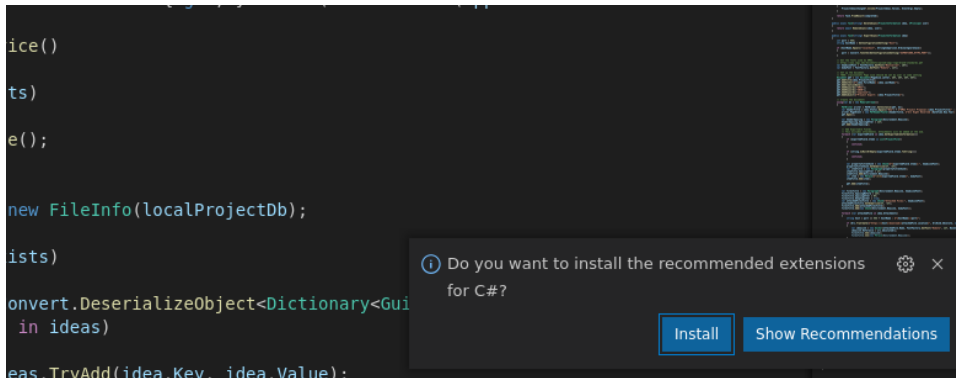


Figure 18: VS Code - Load C# Extension

This is all the setup needed while using the ADF container for CaPPMS. Please note that attempting to debug while using ADF on a Windows machine may causes errors or other failures as nested virtualization depends greatly on the hardware capabilities of the system. It is recommended that Windows users continue to the next section for a delightful experience.

Programming CaPPMS with Windows and Visual Studios

So, you would rather use Microsoft Windows platform to develop in? Visual Studios provides a community edition that is free to use and is best suited for this project. Visual Studios can be a bit daunting and heavy to use though. Be sure to have roughly 15GB of free space on the desired machine to get started.

Getting Visual Studio

- Download the community edition of Visual Studio from:
<https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&rel=16>
- Open the installer
 - The installer may require an update. Hit next if so to allow the update to download
- There are two packages to select in order to get started
 - First, Asp.Net and web development

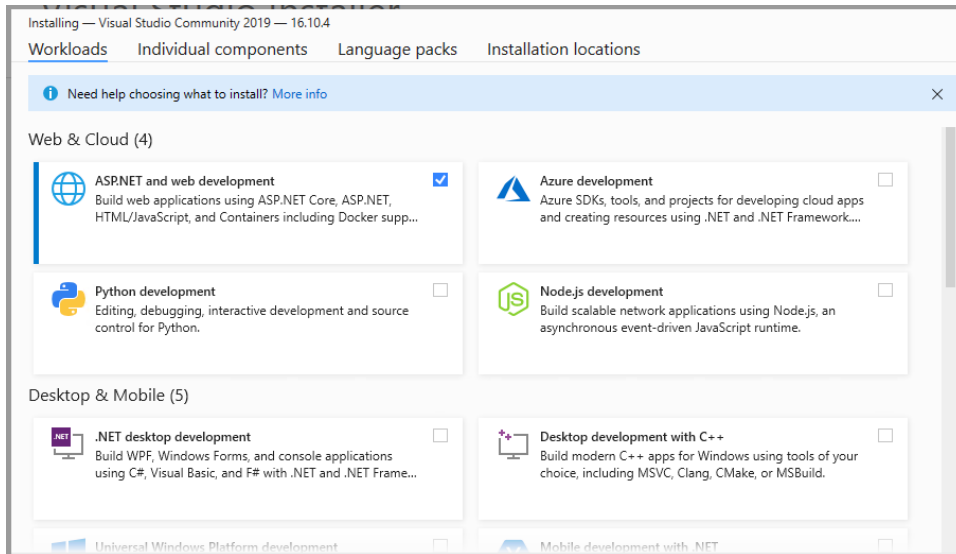


Figure 19: VS Setup - Select Asp.Net

- Second, .NET cross-platform development

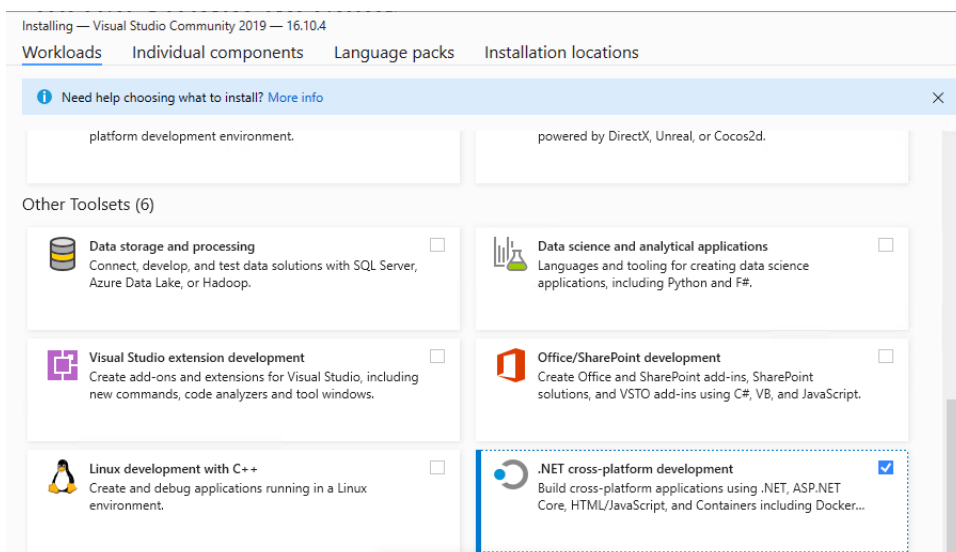


Figure 20: VS Setup - Select .Net cross-platform development

- Click the install button continue
- After installation has completed, Select your general development setting and theme

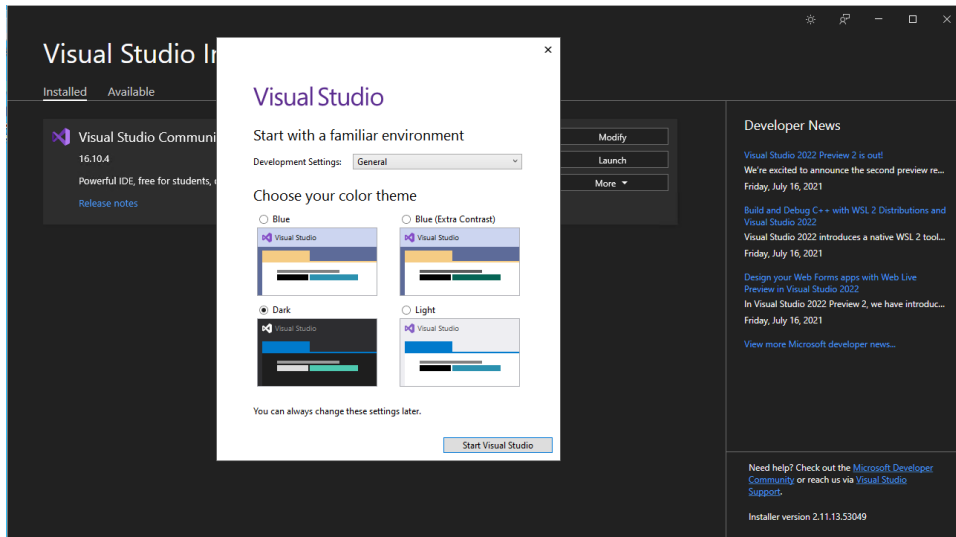


Figure 21: VS Setup - Customize Visual Experience

- Start Visual Studio to validate the installation and any settings that you personally want to cover
 - Pro-tip: At this point Git should already be installed. Create a shortcut on the desktop pointing to: "C:\Program Files\Git\git-cmd.exe"
 - Once the shortcut is created, open the properties and set "Start In:" to: %USERPROFILE%\source\repos
- `cd %USERPROFILE%\source\repos`
- Clone the Repository currently located at:
<https://github.com/umgc/DevSecOpsPlayground.git>



Figure 22: Git Clone Command

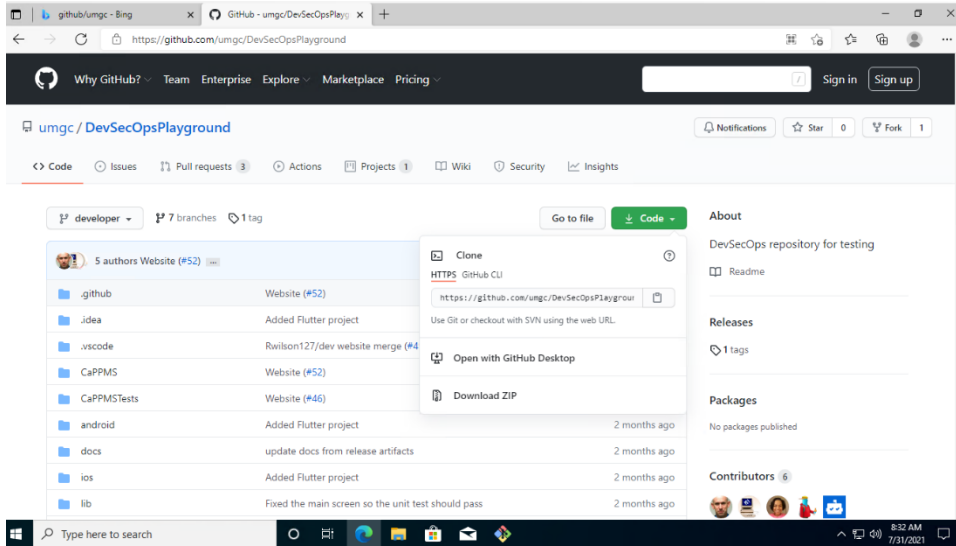


Figure 23: GitHub Code - Clone URL Location

- Run the following commands to get started in CaPPMS or via the file explorer navigate to DevSecOpsPlayground\CaPPMS and open the CaPPMS.sln file
 - `cd DevSecOpsPlayground\CaPPMS`
 - `CaPPMS.sln`

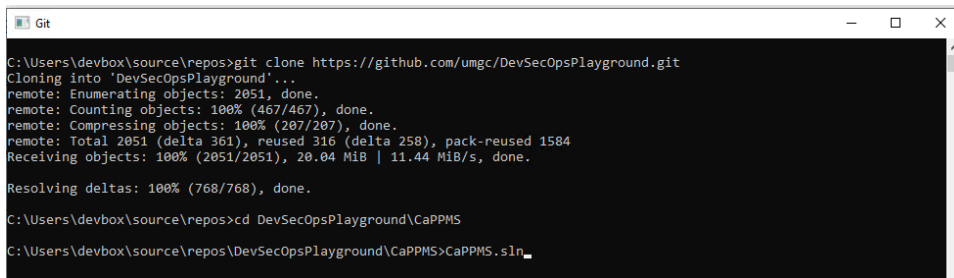


Figure 24: Load CaPPMS from Command Window

- You may have a popup window asking how you want to open the *.sln file. Please select Visual Studio 2019 as shown below

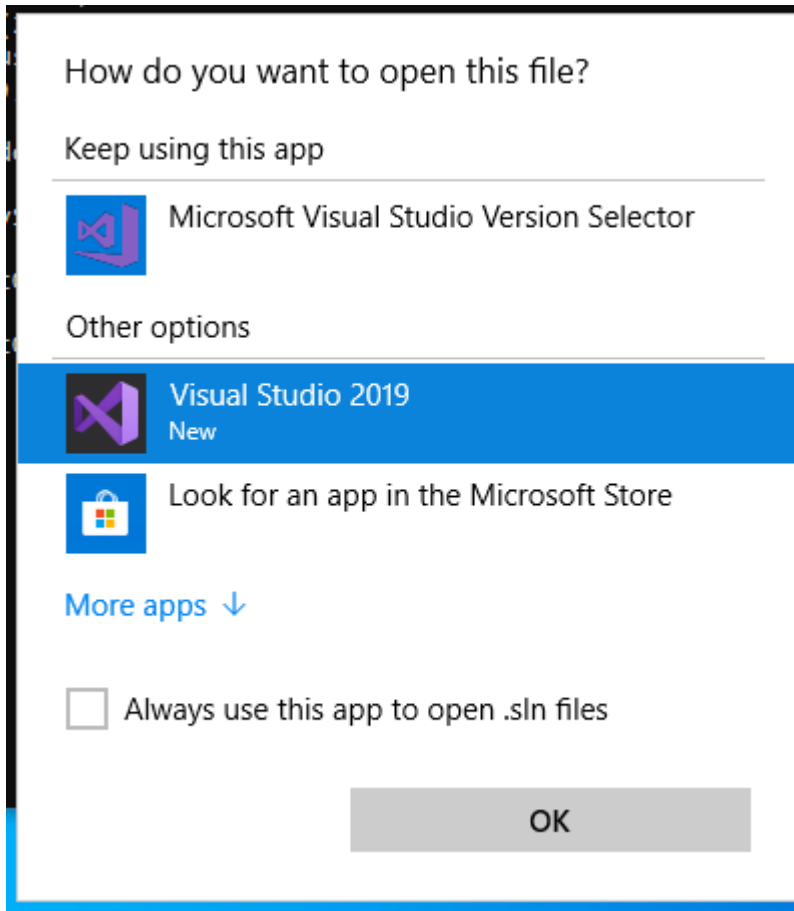


Figure 25: Application Selector

- If Git has not been used before. Please set the user's name and emails address using your own information. The displayed data is only for example and should not be used

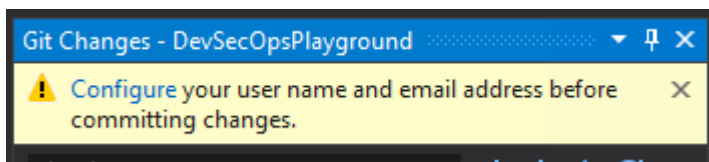


Figure 26: Git Changes Prompt

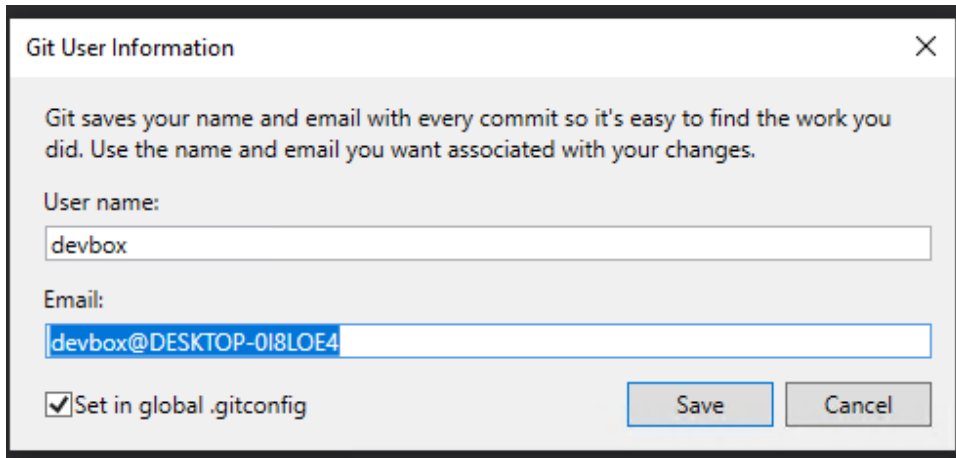


Figure 27: Git User Information Prompt

- Visual Studio will open and load the CaPPMS project. Once loaded you should be able to click the play button

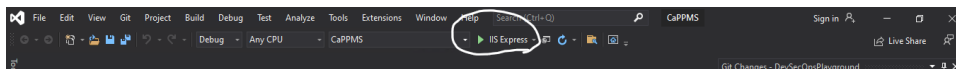


Figure 28: VS Start Debug on IIS Express

- Upon the first start of the project, you may be prompted for SSL/TLS certificate creation. This normal and should be allowed. Installation of the certificate in the following step should also be allowed

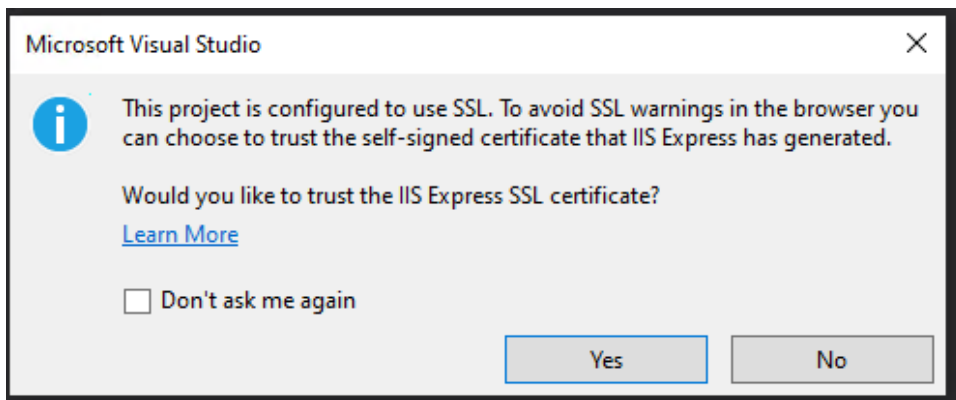


Figure 29: SSL Configuration Prompt

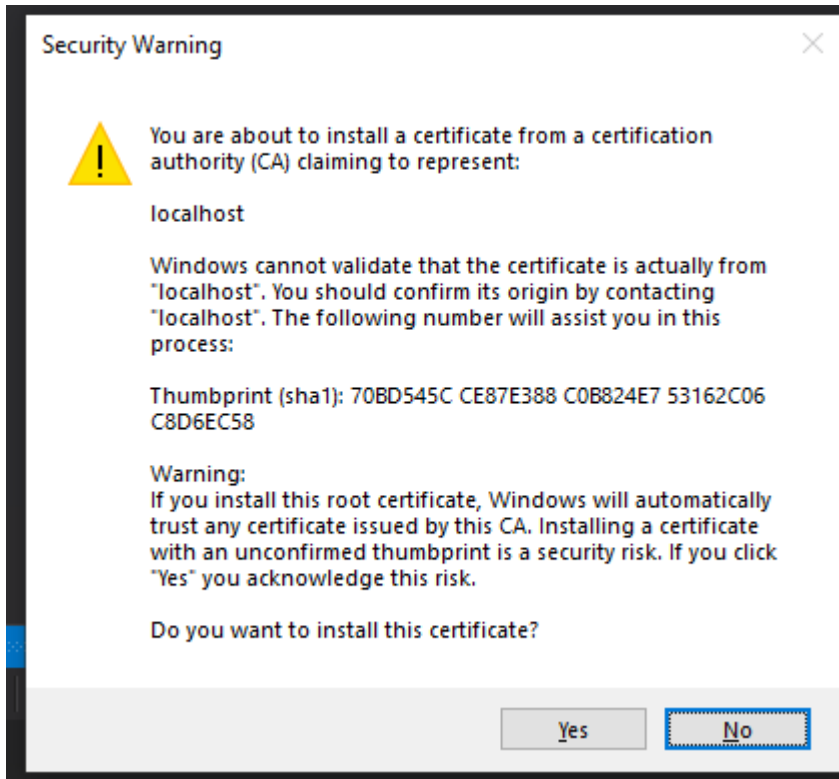


Figure 30: SSL Certificate Install Prompt

- The CaPPMS Home Page should load. This validates that the installation of the project can be loaded and started on the machine

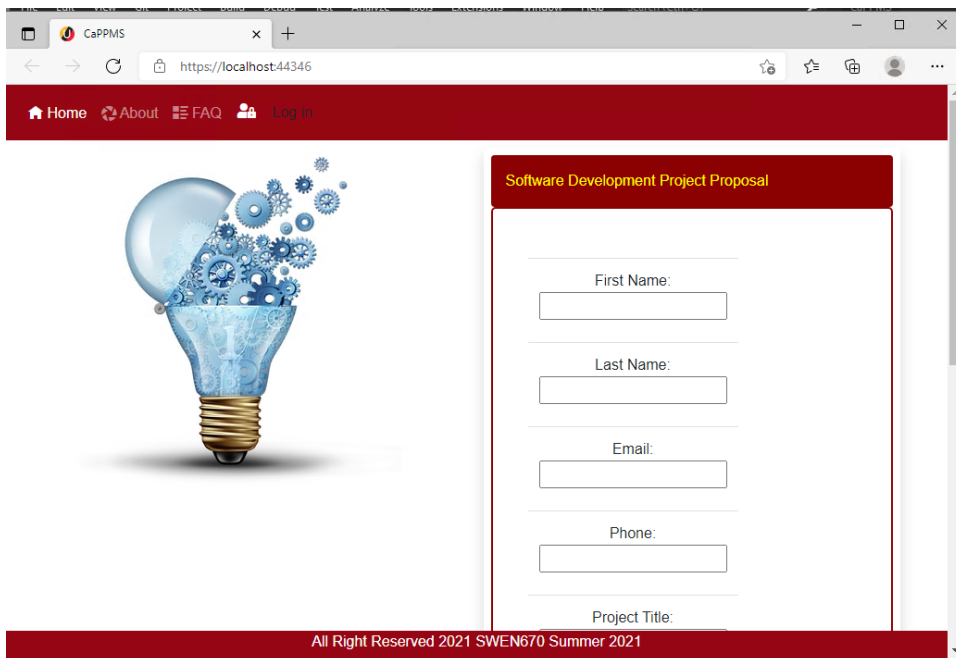


Figure 31: Debug CaPPMS

Developing for CaPPMS

CaPPMS was developed using a relatively new technology stack provided by ASP.Net. This stack is called Blazor. A Blazor application can be either server side or web assembly (client). Server-side Blazor executes the commands of the code on server-side. Likewise, web assembly is a client-side implementation executing the code on the client in the browser.

Since this project is based on ASP.Net here is a helpful link to common coding conventions: [C# Coding Conventions | Microsoft Docs](#).

By this point you are asking yourself what is the big deal? Blazor application generally do not use JavaScript to provide dynamic interactions. This means the complexity of frameworks is greatly reduced. CaPPMS for example was developed without any JS written specifically for it. All the code is in C# which executes and uses the built-in event handling to perform callbacks as needed. The application stack below provides the visual representation of the application layers. For the most part, the orange box is where most of the development takes place with minor exceptions. Services are one of the exceptions. Services are created as Singletons in the Startup.cs file and injected where needed in *.razor components.

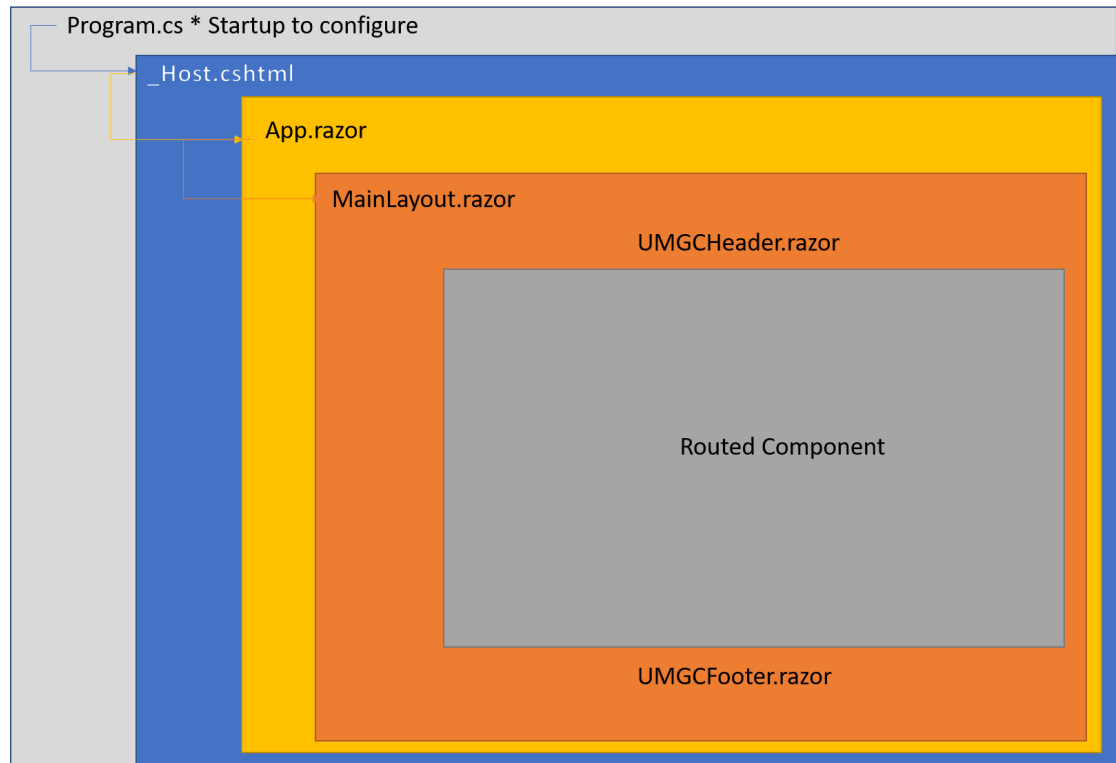


Figure 32: Application Order

File Structure

The file structure has been adapted from the boilerplate structure. Reusable components generally live in the Shared folder and Page components generally live in the Pages folder. Outside of these two folders there are wwwroot, Attributes, Data, Model, and SQL_Creation folders.

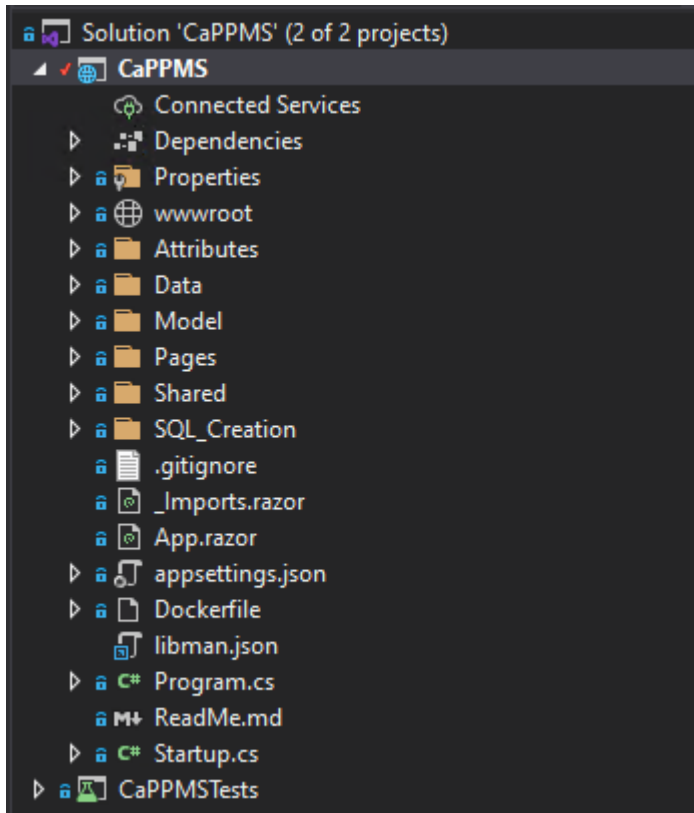


Figure 33: File Structure

The wwwroot folder contains all the usual website related items. This can include typical images resources, favicons, and of course css.

The Attributes folder contains custom attributes used throughout CaPPMS. These include export, spanicon, columnHeader, attachmentsFileSize, and attachmentsNumFiles. These attributes are placed on properties located in Model/ProjectInformation.cs but more about that later.

The Data folder represents two items that develop data. One is the LocalProjectFilesManager. This class is responsible for saving, reading, and deleting attachments as they may be. The other item is ProjectManagerService. This service is registered in Startup and can be injected into all components for consumption. The service is responsible for maintaining project state and related data.

The Model folder houses the model used throughout the project.

Skipping down to the SQL_Creation folder. This folder was created and initial SQL developed by due to the small scale and scope of this project there are no databases in use at this time.

These files are unmaintained and would need to be updated based on current models before put into action.

Package Manager

The package manager used for this project is Nuget. The Nuget package manager comes pre-loaded with Visual Studio (VS) and can be found by navigating: Tools -> Nuget Package Manager -> Manage Nuget Packages for Solution.

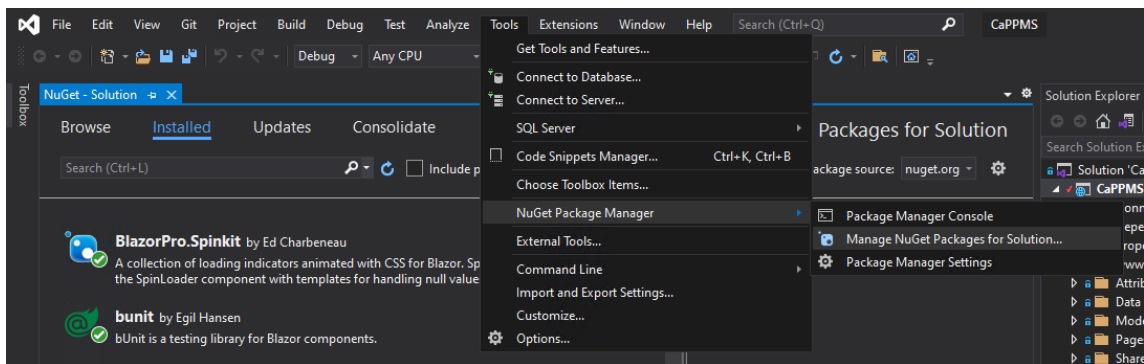


Figure 34: VS Tools - Nuget Manager

Nuget.org is the source repository in use and is preconfigured in VS. To add additional sources, you can click the cog/setting icon in the top right of the Nuget Manager.

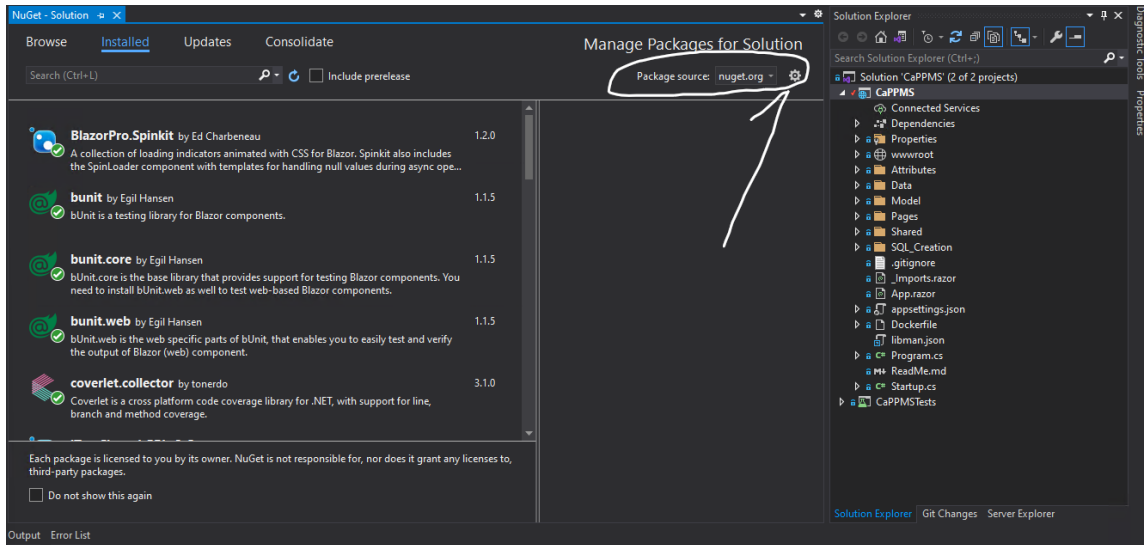


Figure 35: VS Nuget Manager

Using Attributes

The ProjectInformation class uses [System.ComponentModel](#), [System.ComponentModel.DataAnnotations](#), [System.Reflection](#), and [CaPPMS.Attributes](#) as a means to control how the information is rendered to the user and validated when the user submits data back to the webserver. As an example, the property named “ProjectTitle” will be used.

```
[Export(true)]
[Required]
[StringLength(25, ErrorMessage = "Title is either too short or too long. We have confidence you can figure out which.", MinimumLength = 5)]
[DisplayName("Project Title")]
[Browsable(true)]
[ColumnHeader]
16 references
public string ProjectTitle
{
    get
    {
        return this.projectTitle;
    }
    set
    {
        this.projectTitle = value;
        this.IsDirty = true;
    }
}
```

Figure 36: Annotations

Taking a closer look at the attributes we can see the following: Export, Required, StringLength, DisplayName, Browsable, ColumnHeader. Let’s break these down:

- Export allows this property to be exported to a PDF file

- Required enforces this property to have content on form submission
- StringLength defines minimum and maximum string length requirements on form submission
- DisplayName sets the name displayed back to the user. This is used by Table for example to change the column header from the property name to something more human readable
- Browsable exposes the property. More useful in other UI elements but is used here to help describe intent
- ColumnHeader identifies to Table that this property should be used as a column in the table

Let us have a look at another example

```
[Export(true)]
[DisplayName("GitHub")]
[Browsable(true)]
[ColumnHeader]
[SpanIcon("fab fa-github", true)]
4 references
public string Github
{
    get
    {
        return this.gitUrl;
    }
    set
    {
        this.gitUrl = value;
        this.IsDirty = true;
    }
}
```

Figure 37: SpanIcon Attribute Example

- This Property has the SpanIcon attribute. This attribute has two purposes. The first is to tell table to add a span element with the class defined to the cell on output, and to use the span as a link. The below is the result

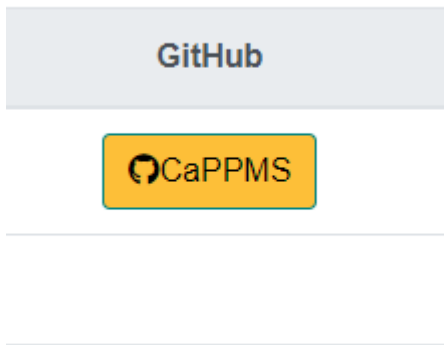


Figure 38: SpanIcon Example - GitHub as Button

- When compared to setting the second argument to false:

```
[ColumnInfoHeader]  
[SpanIcon("fab fa-github", false)]  
4 references
```

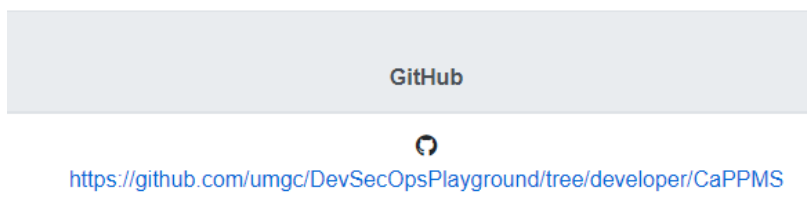


Figure 39: SpanIcon Example - GitHub as Anchor

Implementing an attribute requires reflection. Reflection is a technique to ask the object to find metadata about itself. This metadata can be used for decision making in this case. Let's have a look deeper into this spanicon attribute as the Table implements it. Under Model/Table namespace there is a class called Cell. Around line 84 of the class, starts the implementation of SpanIconAttribute. Here is a screenshot taken at the time of writing:

```

SpanIconAttribute spanIcon = Attributes.FirstOrDefault(a => a.GetType() == typeof(SpanIconAttribute)) as SpanIconAttribute;
bool handledValue = false;

if (spanIcon != null && !string.IsNullOrEmpty(spanIcon.IconClass) & !string.IsNullOrEmpty(cellData))
{
    string spanData = string.Empty;
    if (spanIcon.UseAsHyperLink)
    {
        spanData += $"<a class=\"btn btn-secondary\" href=\"{cellData}\">";
        handledValue = true;
    }

    spanData += $"<span class=\"{spanIcon.IconClass}\"></span>";

    if (spanIcon.UseAsHyperLink)
    {
        spanData += $"{cellData.Split(new char[] { '/' }, StringSplitOptions.RemoveEmptyEntries).Last()}</a>";
    }

    html += spanData;
}

```

Figure 40: SpanIcon - Implementation

The first line attempts to ask a cell property, Attributes, for any attribute with the type SpanIconAttribute then casts the returned object to SpanIconAttribute. This type of casting in C# is pattern matching and can yield a null value. So the first condition to check is null and then we check in this example some other items to make sure there is actionable data. After that we check the properties of the spanIcon to get how the span should be classed and if it should be wrapped in a link or not. If wrapped in a link, it will be rendered as in the first picture above that looks like a button. If not wrapped in a link, then the rest of the data is handled later in the code and just the span is added.

Page Flow

Not every page serves the same purpose, and therefore, expecting every page to look the same is unrealistic. At the point you only need one page. Now is a good time to walk through an example of a page. The projectlist page will be used for the example. The page is located in the Page folder with the filename of projectlist.razor. This section will attempt to tie the concepts of the page, cascading parameters, parameters, and components together. The projectlist page is also a component so let us start there:

```

1 @using CaPPMS.Data
2 @using CaPPMS.Model
3 @inject ProjectManagerService ProjectManager
4
5 @page "/projectlist"
6
7 <CascadingAuthenticationState>
8     <ProjectTable DataSource="new List<ProjectInformation>(ProjectManager.ProjectIdeas.Values.OrderBy(i => i.Status).ToList())" CssClass="table border p-2" />
9 </CascadingAuthenticationState>
10
11 @code {
12
13

```

Figure 41: Component - Cascading Parameters

- `@using CaPPMS.Data` and `@using CaPPMS.Model` are similar to import or include statements found in other languages and help bring namespaces into scope
- `@inject` is used to inject the `ProjectManagerService` into the page for consumption
- `@page "/projectlist"` is used by the router to display this page instead of some other page
- `<CascadingAuthenticationState></CascadingAuthenticationState>` wraps an item to pass authentication object down to. As a matter of thought, Cascading can be thought of as a waterfall that pours into another waterfall that pours into another waterfall...basically cascading items allow the passing of parameters to nested objects easily. In this case we are passing the authentication state of the page to the nested component: `ProjectTable`. As can be seen, the `ProjectTable` can accept two parameters. The first parameter in is the `datasource` which is getting filled by the injected service. The second parameter is `css` class statement the table should use when rendering.

The page is pretty empty after the only component (element) that should be rendered. There is an empty `@code` section at the bottom of the page. The `@code` section can be found on most `*.razor` pages as a point to embed C# code. More on this in the `ProjectTable` component.

The starting point of this example is to look at the `@code` section of the `ProjectTable` component that can be found in the `Shared/ProjectTable.razor` file.

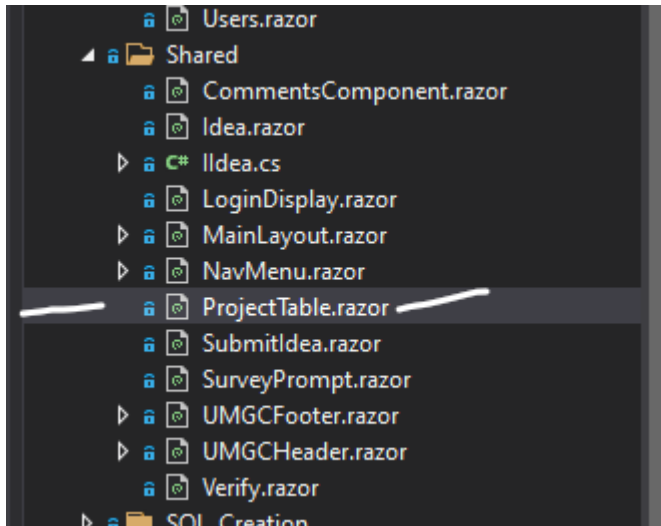


Figure 42: File Structure – ProjectTable.razor Call Out

This is easily the most complex component in the project in terms of structures. The Idea.razor component is the most complex component in terms of rendering and as the reader, you should take some time to go over that later. It is basically an html page that will render sections based on conditions. Back to this component though.

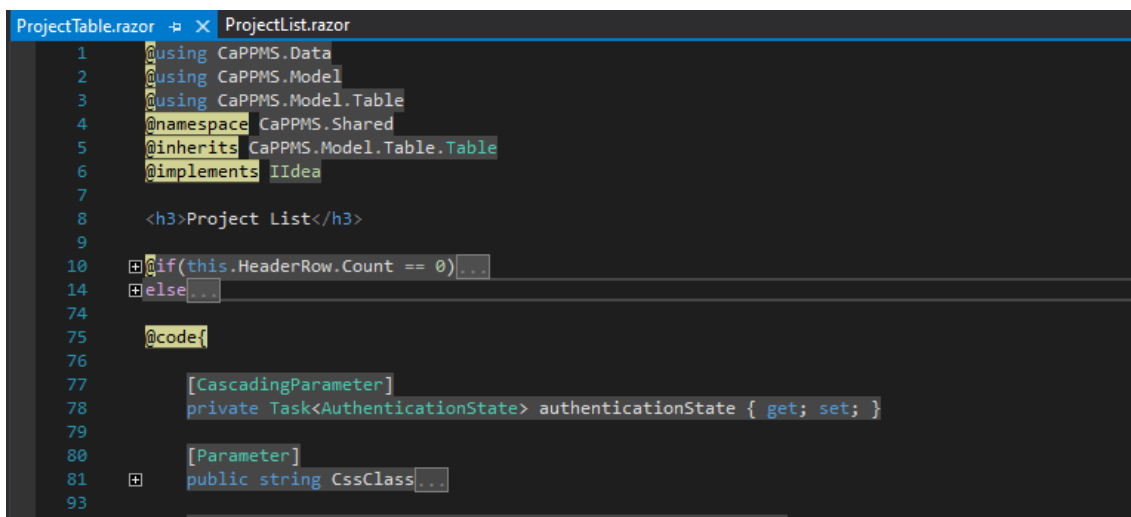
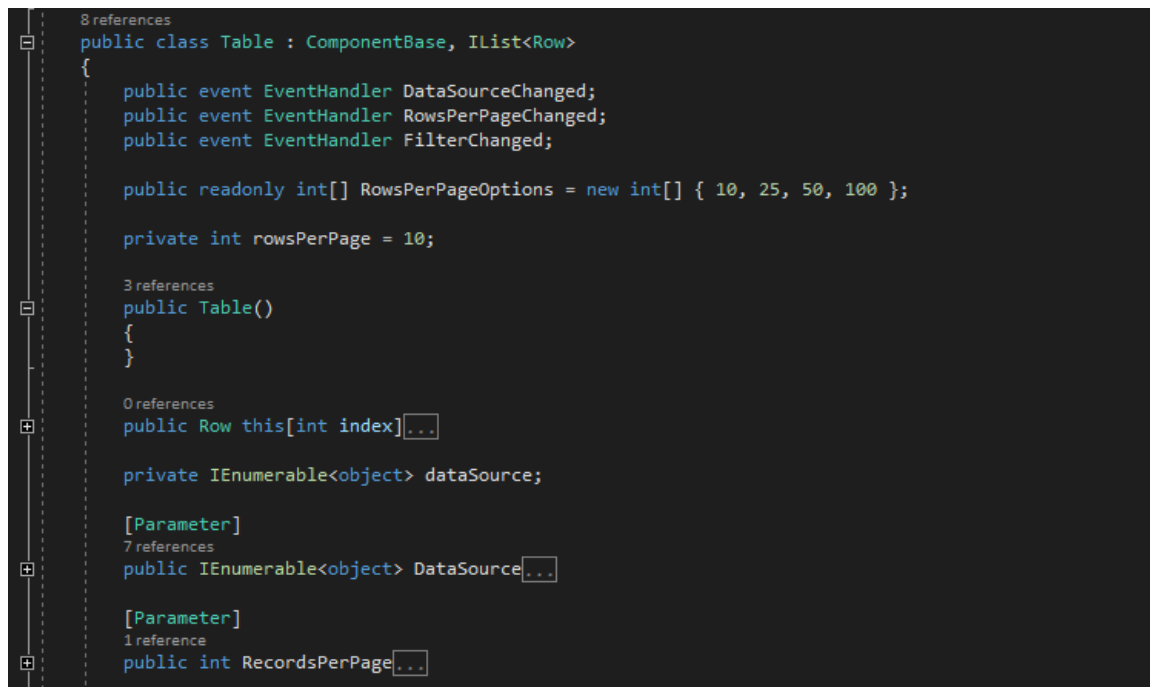


Figure 43: Component - Directives

The meat of the rendered page is collapsed for brevity. There are some important items to see right away here. First is that this component inherits from some other class. The second it that it implements an interface. The third is that we passed in two parameters and one cascading

parameter but annotations for the cascading and just one parameter can be found. The DataSource parameter is in the Table class this component inherits from.



```
8 references
public class Table : ComponentBase, IList<Row>
{
    public event EventHandler DataSourceChanged;
    public event EventHandler RowsPerPageChanged;
    public event EventHandler FilterChanged;

    public readonly int[] RowsPerPageOptions = new int[] { 10, 25, 50, 100 };

    private int rowsPerPage = 10;

    3 references
    public Table()
    {
    }

    0 references
    public Row this[int index] { .. }

    private IEnumerable<object> dataSource;

    [Parameter]
    7 references
    public IEnumerable<object> DataSource { .. }

    [Parameter]
    1 reference
    public int RecordsPerPage { .. }
```

Figure 44: Table Model

From the above picture we can see that DataSource has been annotated with Parameter and actually one more unused parameter is here as well. It is equally important to know that Table is derived from ComponentBase and implements IList<Row>. This allows the entire table to be enumerated easily calling each row if the consumer wants. RowsPerPage was not implemented during this development cycle as time was a constraint and this feature was a low priority. So now back to the purpose of the document, the markup to render for the viewer!

