# DevSecOps
# User Guide

Michelle Monfort
Jeroen Soeurt
Robert Wilson

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

Table of Contents

Table of Figures:

## Introduction

The DevSecOps team has completed several projects during the summer semester. The deliverables of these projects can help current and future developers in the UMGC software engineering capstone course in their work. They include and containerized development environment (Advanced Development Factory), and a website to manage project ideas and suggestions.

The other part of the DevSecOps team's responsibilities included setting up and maintaining the git repositories for the development teams on GitHub.

This user guide will cover all three areas. It starts with the use of Git, then covers the ADF, and finally describes the use of the website.

Git

## Introduction

Git is a version control system originally developed by Linus Torvalds in the mid-2000s. It is free and open source, and is used by online repository providers like GitHub, Azure DevOps, and BitBucket. For this course we will be using git in combination with GitHub. GitHub provides free and paid repositories. As a UMGC student you can get the GitHub Student Developer Pack, which gives you a pro account and several software products and services for free. More information about this can be found at https://education.github.com/pack.

## 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.

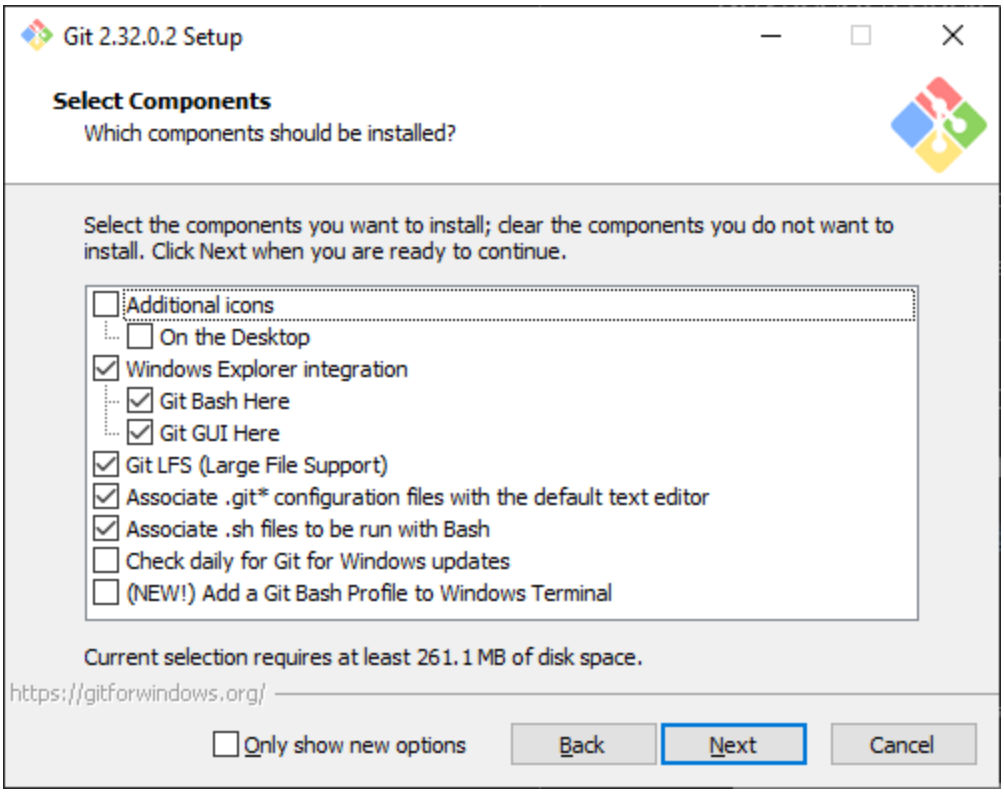



*Figure 2: GIT Setup Screen 2 - Common Options*

Click next several times until you get to this screen:

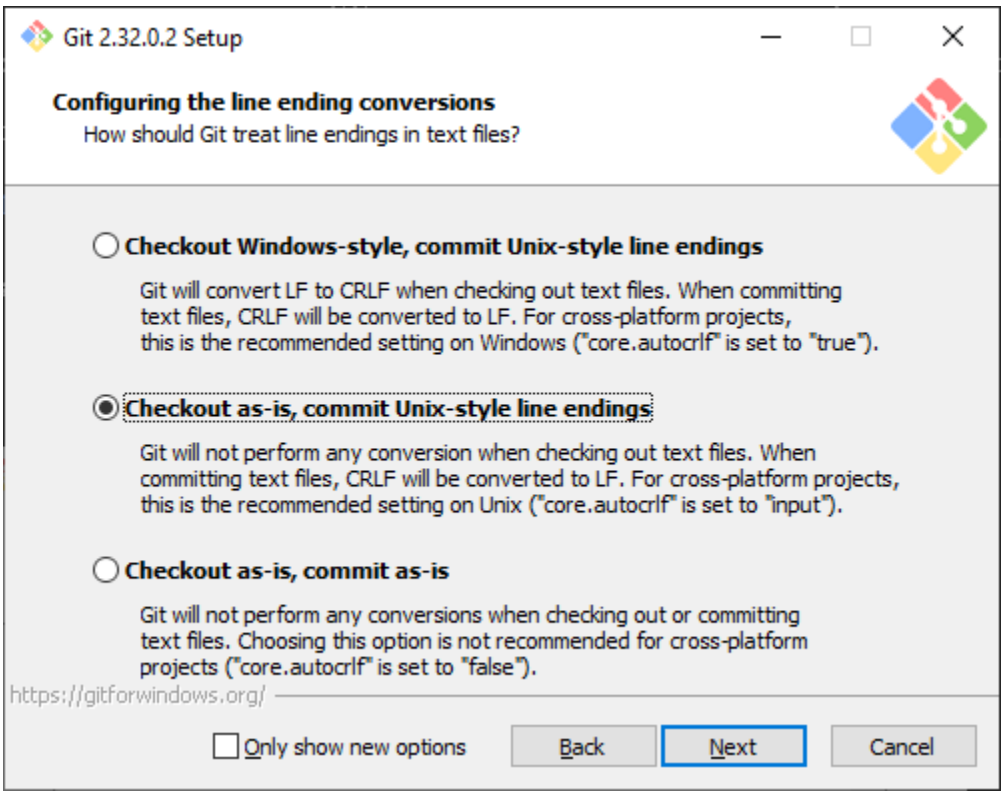



*Figure 3: : GIT Setup Screen 3 - Use Unix style line endings*

Here it is important to select Checkout as-is. Otherwise, you run into problems when cloning the ADF in the next chapter. 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**

The DevSecOps team will provide you with access to the UMGC capstone organization in GitHub (https://github.com/umgc/). Here you will find repositories from previous terms. DevSecOps has also created a repository this term's teams. Navigate to your team's repository to get started.

### Cloning

On your repository, click on the green Code button, and copy the URL listed.

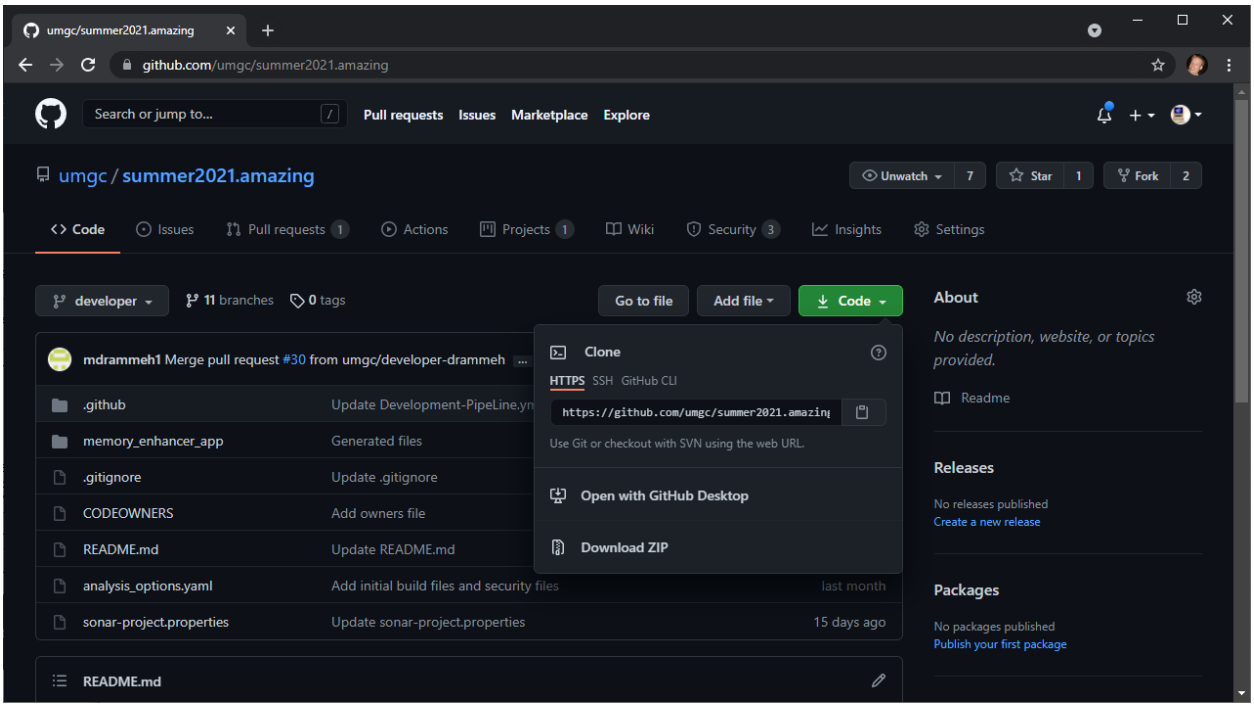



*Figure 4: GitHub Clone Repository Url*

Now open a new terminal window, and clone the repository using the following command:

- git clone <url>

Example:

- git clone https://github.com/umgc/summer2021.amazing.git

This will create a new folder with the new of the repository, and download the repository.

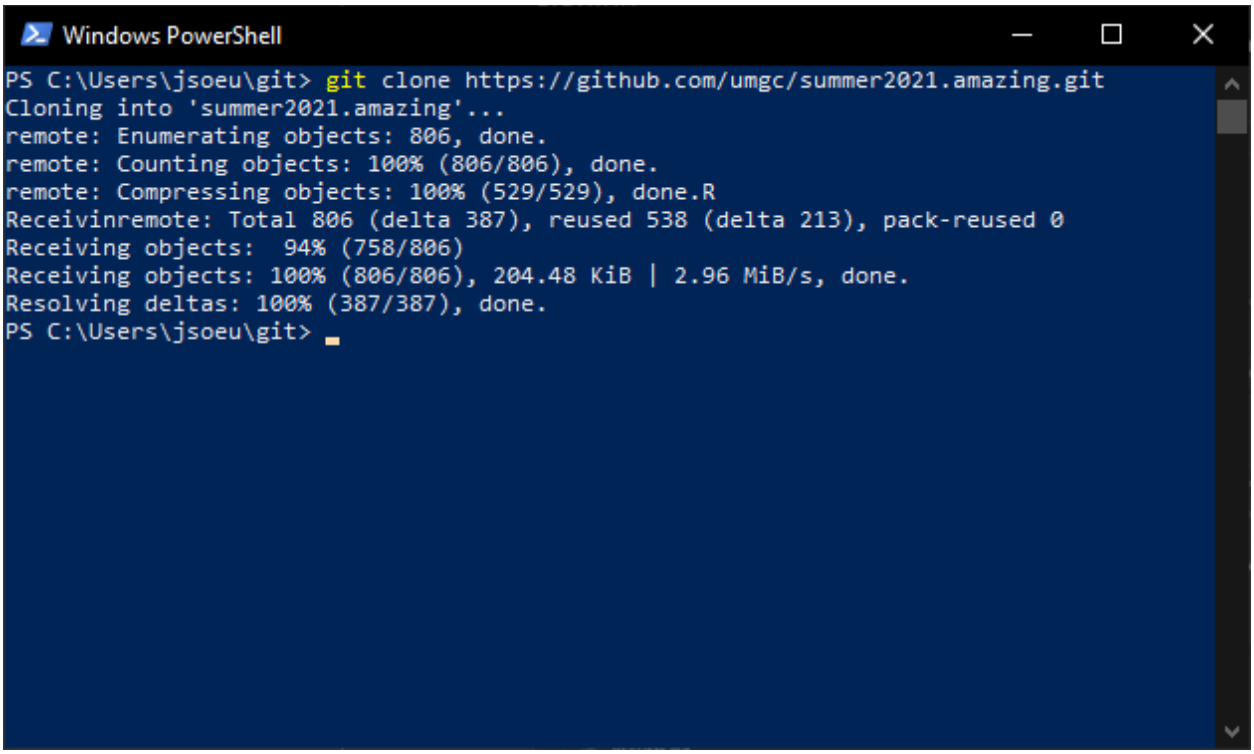



*Figure 5: Command Line - Clone Command*

**Branches**

Git uses the concept of branches. You can create a new branch using:

- git checkout -b <branchname>

Example:

- git checkout -b jersoe/encryption

This creates a new local branch. You can make changes to the code. You can also switch back to other branches using:

- git checkout <branchname>

Example:

- git checkout developer

You generally want to create a new branch for your work. When you are satisfied with your changes, you have to perform a few steps:

*Staging*

First, you have to stage the file that you have changed. This can be a specific file, or all files that have changed.

All files:

- git add .

A specific file:

- git add crypto.java

*Commit*

Next, you have to commit your changes. To do so, use the following command:

- git commit -m “<message>”

Example:

- git commit -m “Added encryption.”

*Push*

The next step in your workflow is to push your changes to the remote repository on GitHub. Now there are two options. If you created a new local branch, then this branch will not exist on the remote repository. In that case, use:

- git push --set-upstream origin <branch>

Example:

- git push --set-upstream origin jersoe/crypto

If you’ve previously pushed this branch to remote, then the following will suffice:

- git push

*Pull Request*

The final step is to request that your code in merged into the development branch. You can request this by creating a pull request. To do so, navigate to the GitHub repository, click on Pull Requests, and use the green “New pull request” button.

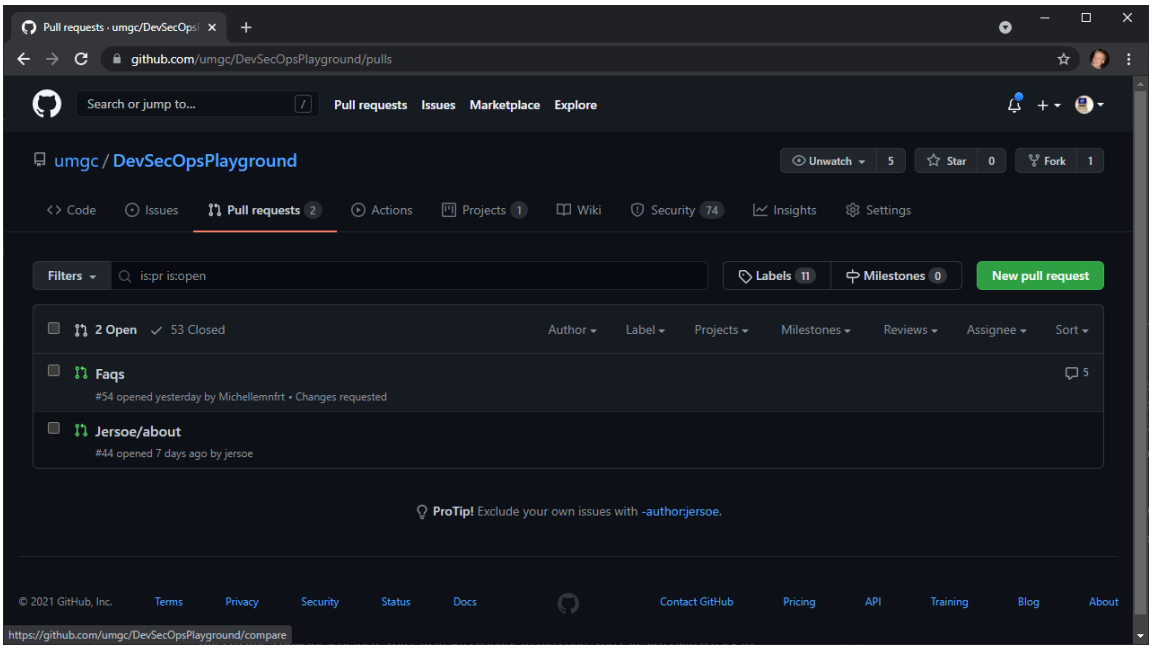



*Figure 6: GitHub Pull Request Creation*

At the top, choose the target branch (developer), and the source branch (your work):

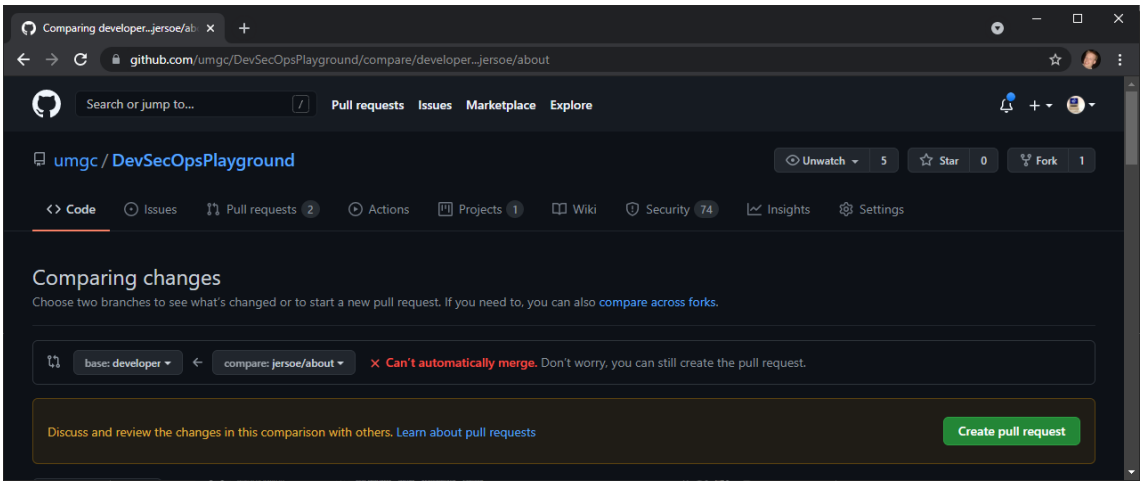



*Figure 7: GitHub Base Branch Selection*

Next click the green Create pull request button.

The pull request will now be reviewed by the project manager of your team, and by DevSecOps.

*Reviewing a pull request*

The repositories have been set up to require reviews. As a project manager or DevSecOps team member you can review open pull requests. To do so, navigate to the repository, click on Pull request, and choose the PR you would like to review.

There are two areas to pay particular attention to: Check, and Files changed.

- Checks show the result of the pipeline that ran. This lets you know if there are any formatting issues, or vulnerabilities for example.

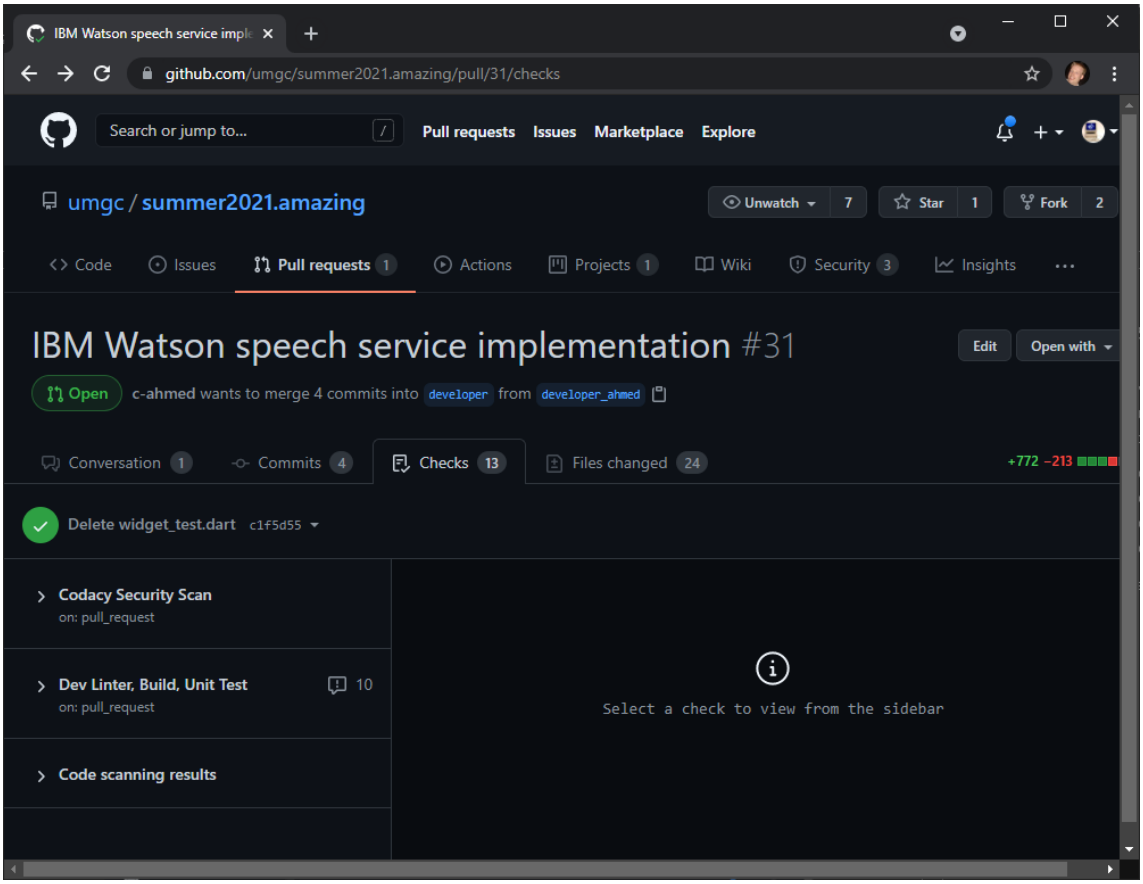



*Figure 8: GitHub Check Status of Pipeline*

- The Files changed tab show all old/new/replaced code. Here, pay attention that there are no issues such as:
    - Commented out code.
    - Left-over boilerplate code.
    - Naming-convention issues.
    - Bad practices.

- Secrets being part of a commit.

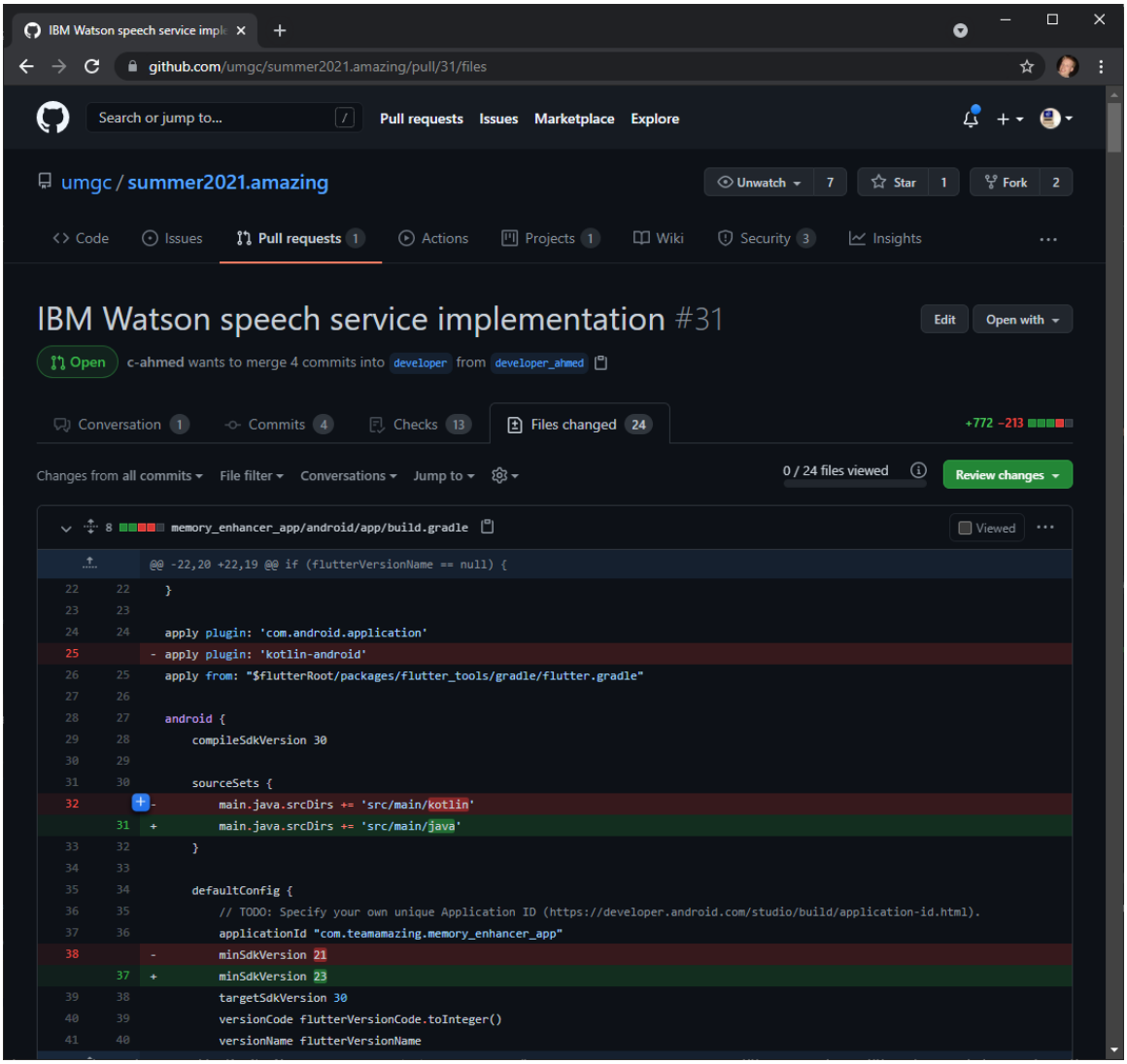



*Figure 9: GitHub Review Files*

As a reviewer, you can approve a PR, comment on a PR, and request changes from the submitter.

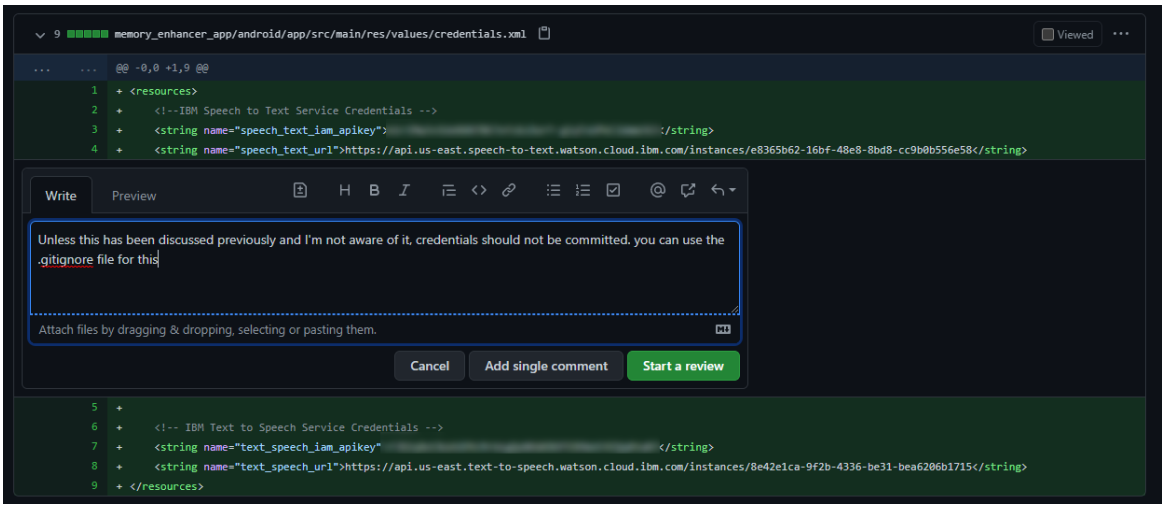



*Figure 10: GitHub Review Files - Insert Comment*

You can add comments to code by clicking the + button in front of a line of code.

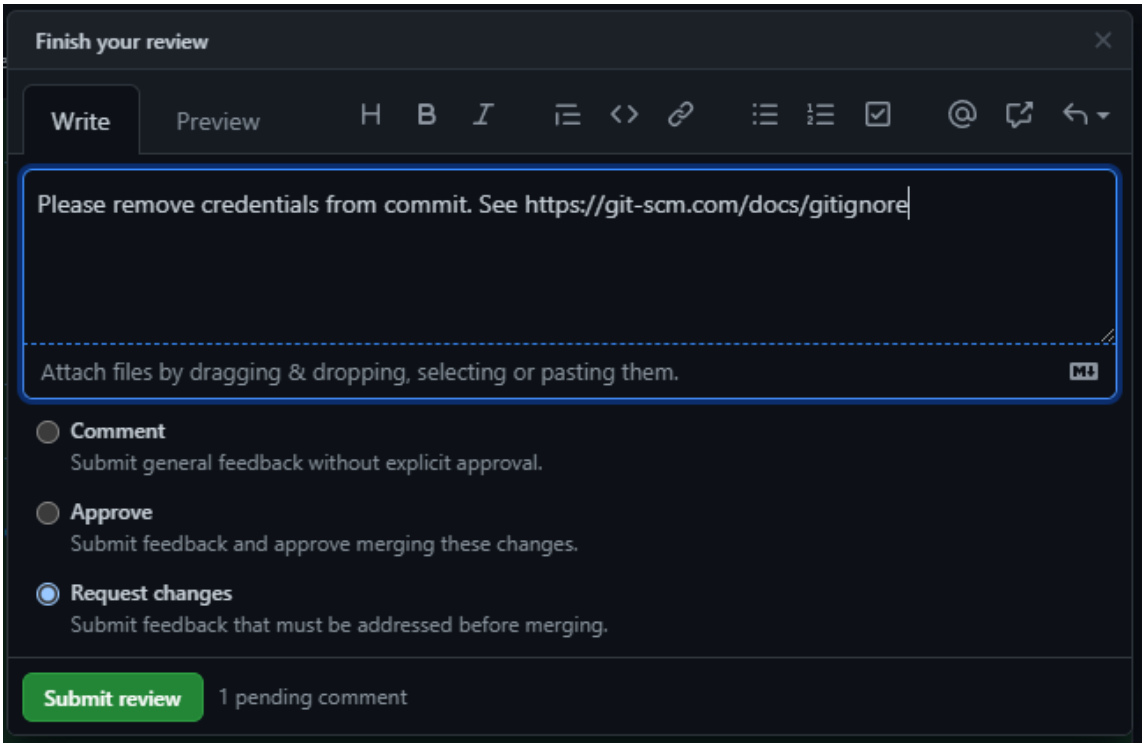



*Figure 11: GitHub Review Files - Request Changes*

Finally, approve the request if you do not see any issues, or request changes if needed.

**Merging**

If a pull request has been approved, you will be able to merge it. To do so, click the green Merge pull request button on the pull request page.

**Resolving merge conflict**

Working on a multiple person team in or around the same area of a project can lead to merge conflicts. This usually happens because two different people editing the same code at nearly the same time and one PR is merged before the other. It may also happen that you have multiple PRs open and they might touch the same code as well, even if in a minor way. Luckily GitHub makes it super easy to resolve these conflicts right in the browser.

Let's take the example where one person added a section to the CaPPMS ReadMe.md file while another person was working on their section of the ReadMe file.

During creation of a PR in GitHub, it is obvious that there is a merge conflict. After selecting the base branch, a strong red not appears letting you know that this cannot be automatically merged.

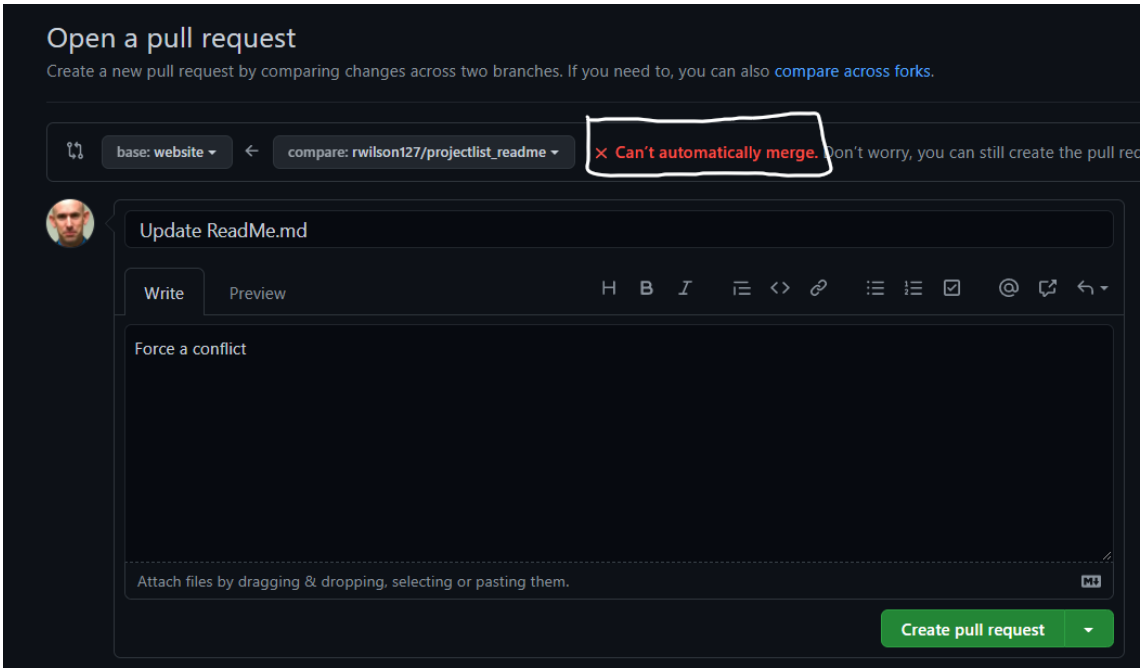



*Figure 12: GitHub Pull Request - Merge Conflict*

- Create the pull request
- Click the Resolve Conflicts button

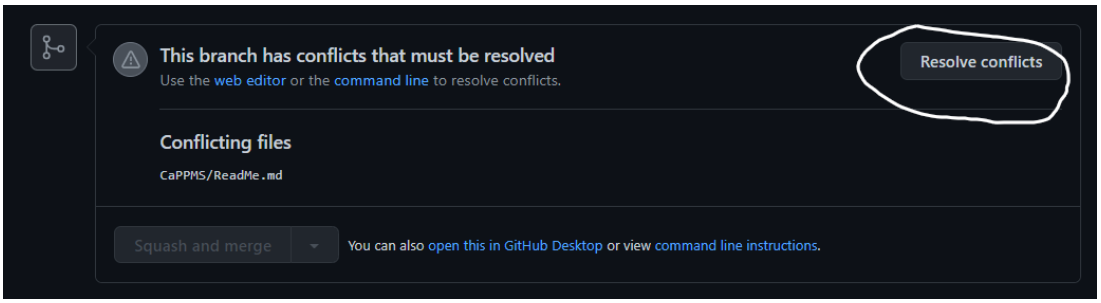



*Figure 13: GitHub Pull Request - Resolve Conflicts Button*

At the top of the page, there are some hints to help you resolve the problems:

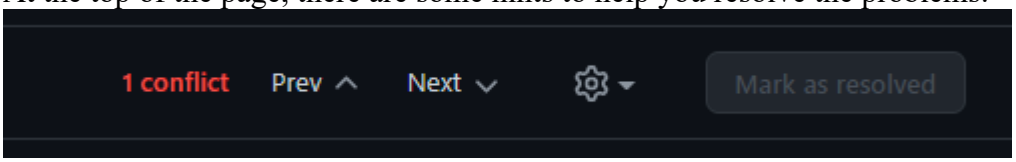



*Figure 14: GitHub Resolve Conflict - Identify Number of Conflicts*

- Some other hints include a line that identifies the area of the conflicted code. In the picture below, the red arrow helps locate this. The yellow underlined section help locate merge markers. These markers help a user understand what code was added on what branch.

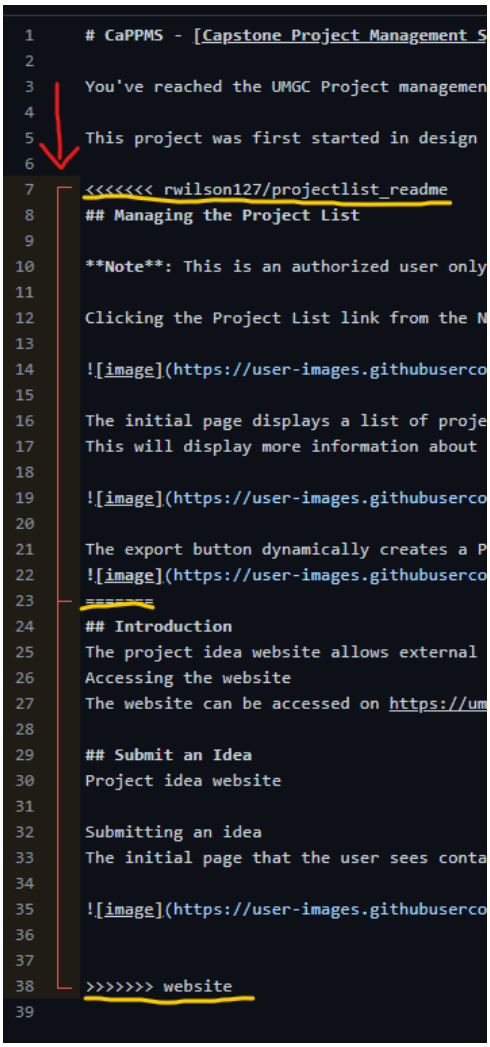



*Figure 15: GitHub Resolve Conflict - Identify Merge Markers*

- Depending on the desired end result, the user might take only from their own branch, the remote branch, or a combination of the two branches. Whatever the case, the merge markers need to be removed before the conflict can be considered to be resolved. In this case we will take from both branches but move the website code above the user branch code section.

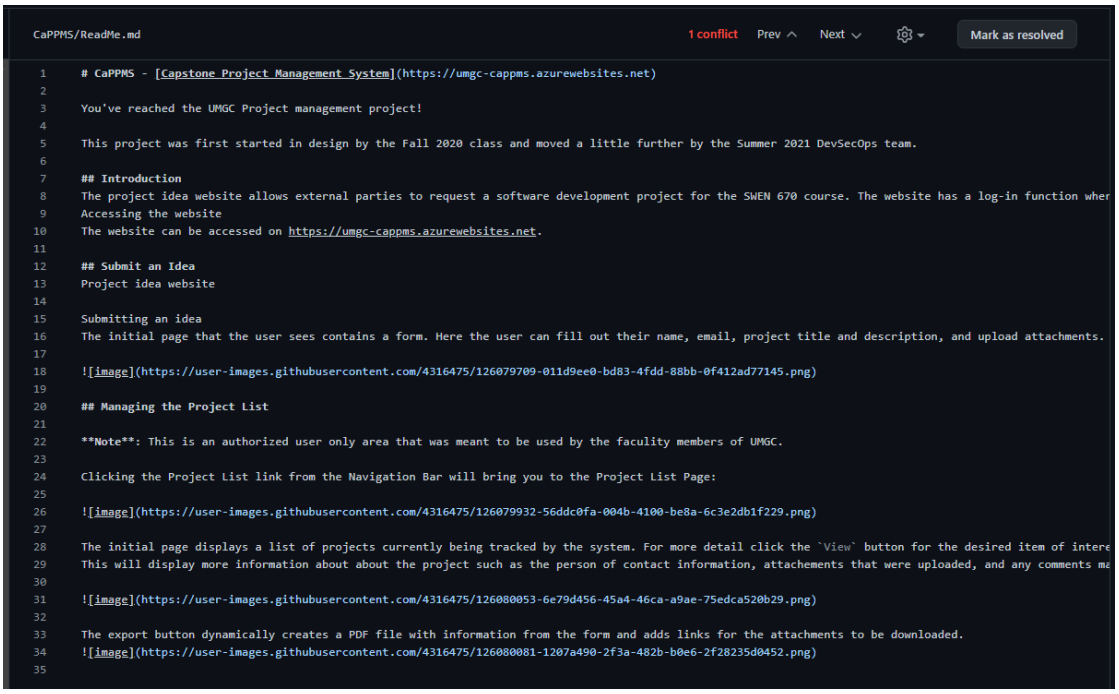



*Figure 16: GitHub Resolve Conflict - Edit document*

- With the code re-arranged and the merge markers removed, the next step is to click "Mark as resolved" in the upper right of the pane. This will result in a green check mark and let us Commit merge.

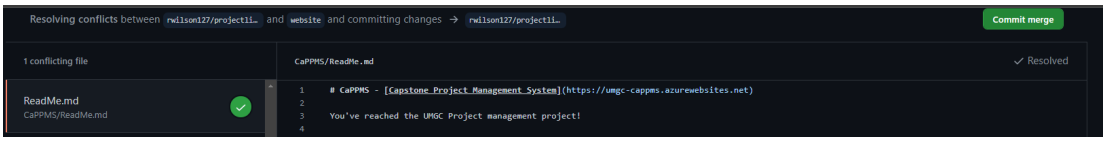



*Figure 17: GitHub Resolve Conflict - Commit Merge*

- The end result is the PR can now be merged onto the desired branch without further conflict.

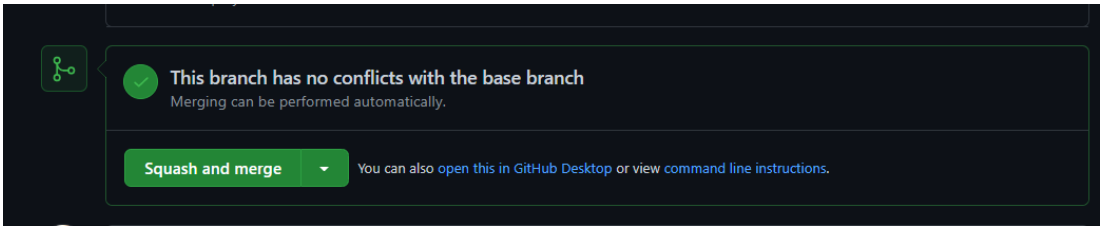



*Figure 18: GitHub Resolve Conflict - Merge Resolved Pull Request*

Pipelines – GitHub Actions

The DevSecOps team has set up several pipelines in your repository. These consist of scripts that run when a certain action occurs, for example when a pull request is submitted. They contain steps to check the code, check formatting, and depending on the branch, deploy your application.

These scripts can be seen in the .github directory in the root of your repository.

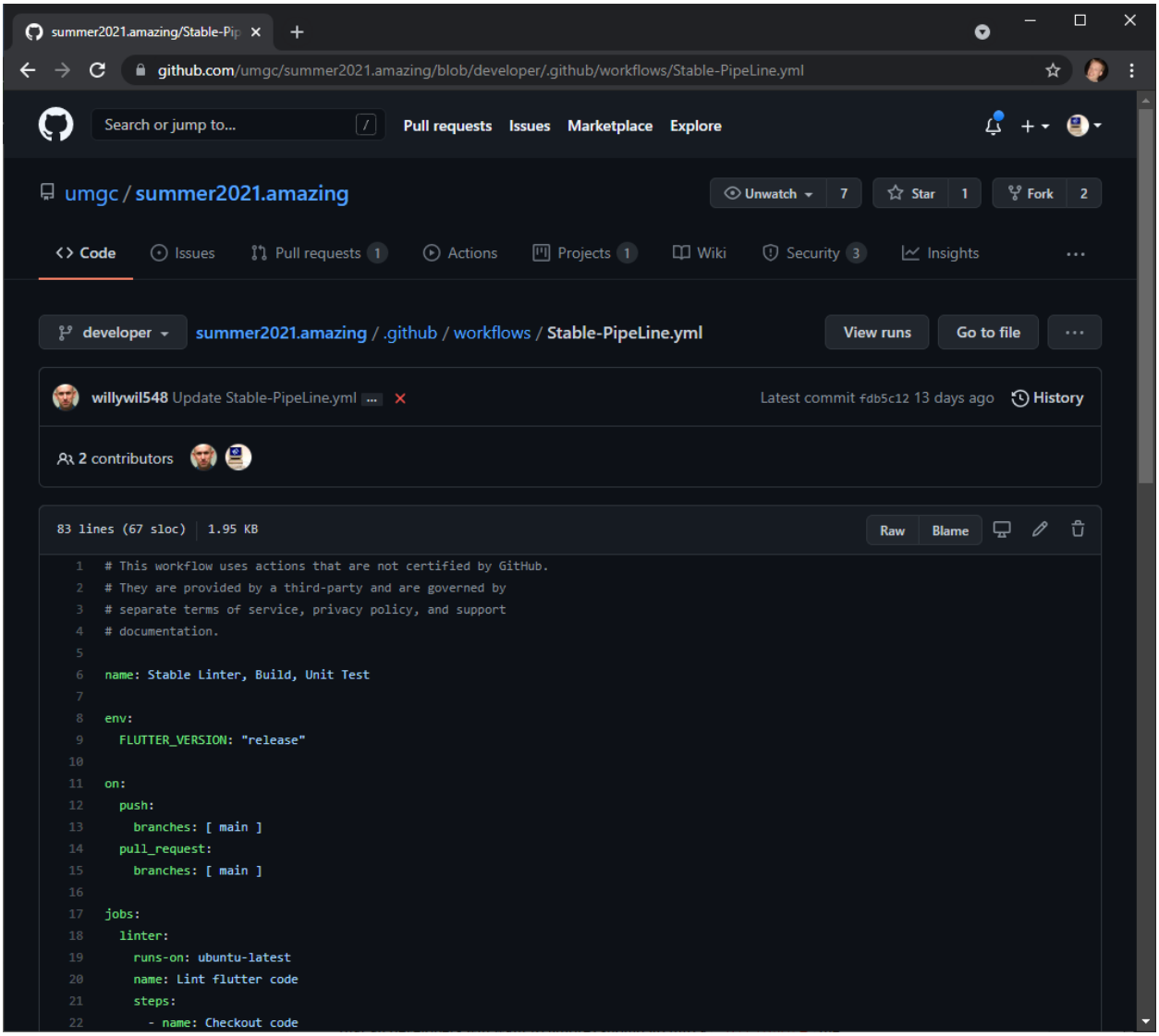



*Figure 19: GitHub Actions - Actions YAML Configuration File*

The result of this pipeline can be seen in the pull request screen under the Checks tab, but also on the main repository screen under Actions.

The pipelines set up currently build your project and store the resulting APK file.

**Retrieving the APK**

To retrieve the APK, navigate to your repository, then Actions, and choose a pipeline.

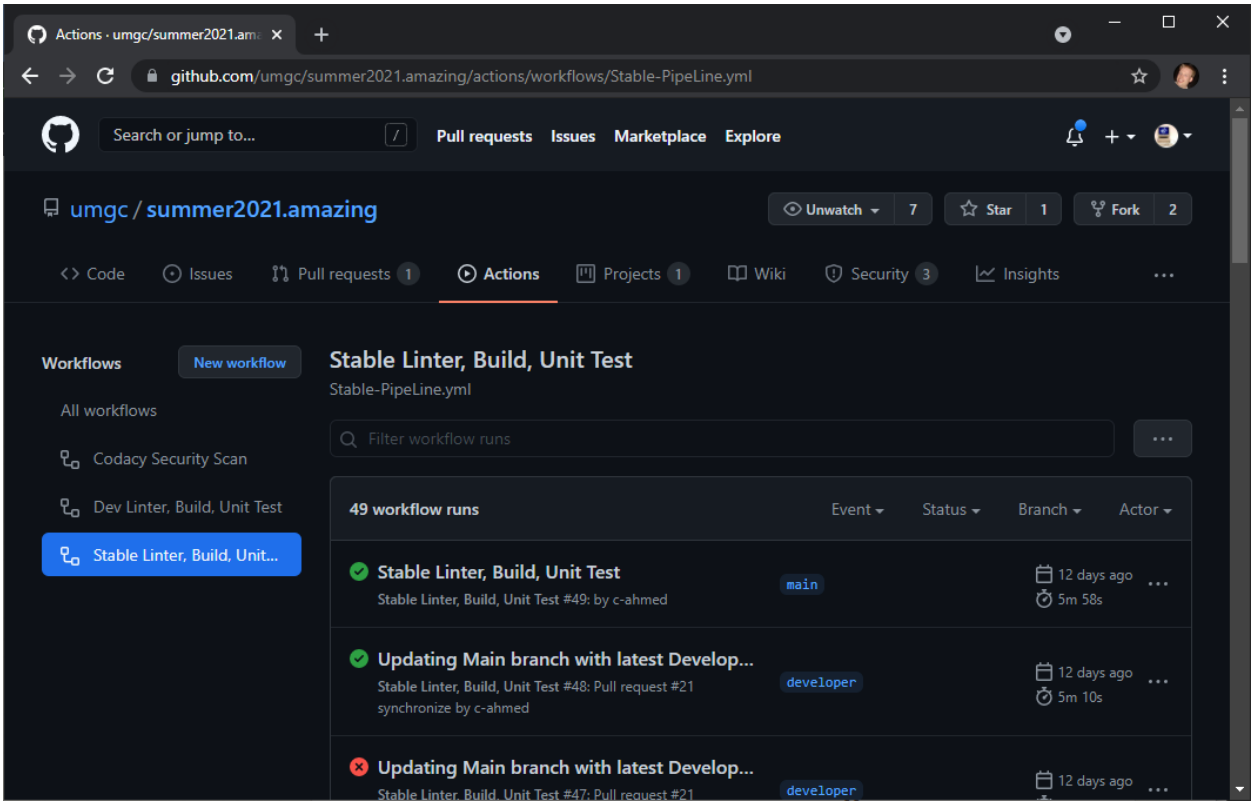



*Figure 20: GitHub Actions - Workflows*

Next, scroll all the way to the bottom of the page. Here, under artifact, you can download the built APK.

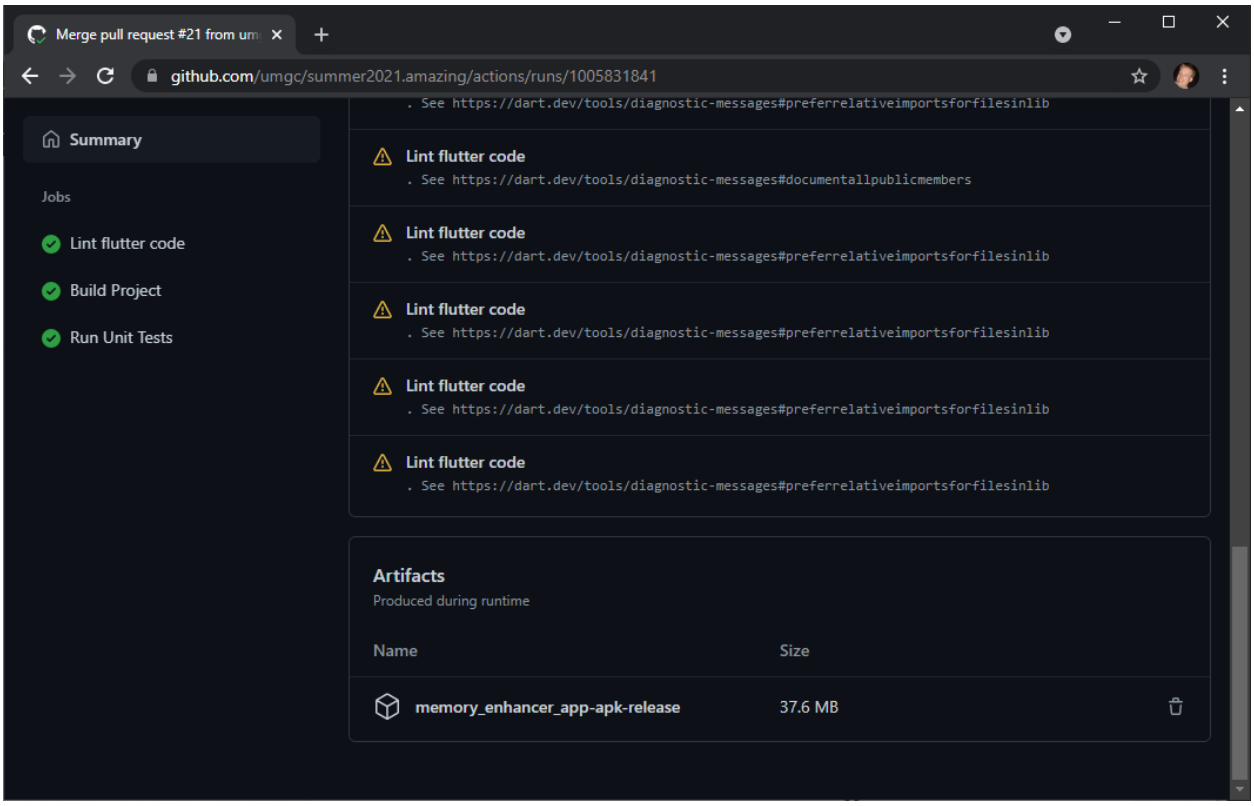



*Figure 21: GitHub Actions - Artifacts*

From here, you can copy the extracted APK to an Android device over USB.

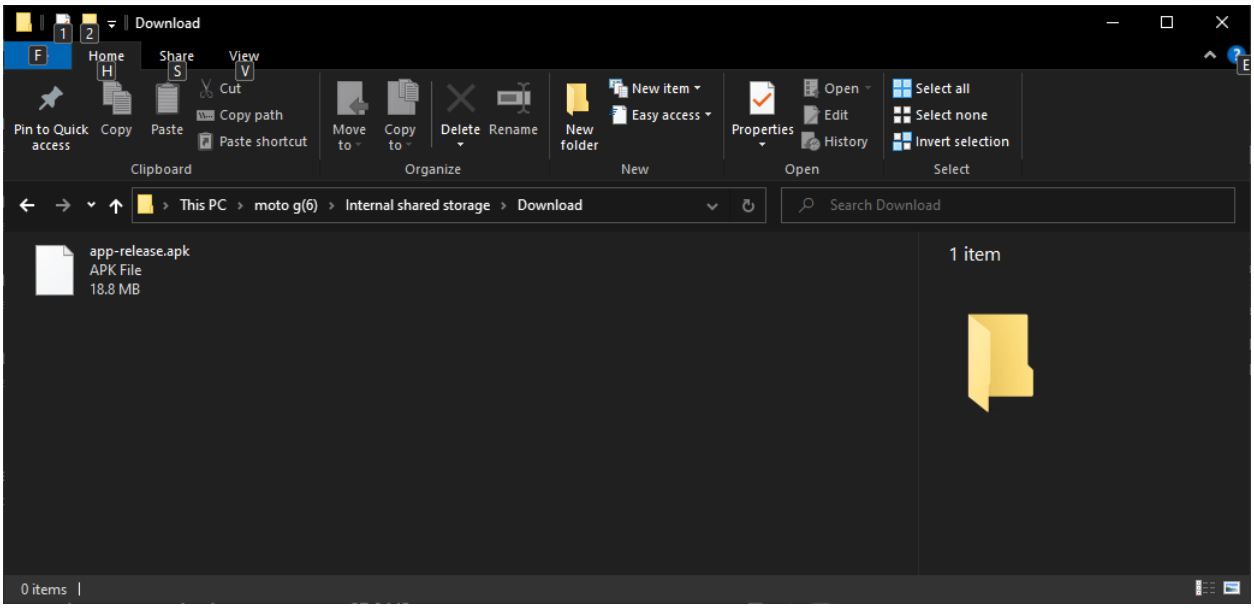



*Figure 22: GitHub Actions - Downloaded Artifact*

Now, on your device, there will be several warnings. This is because the app does not originate in the Google Play store.

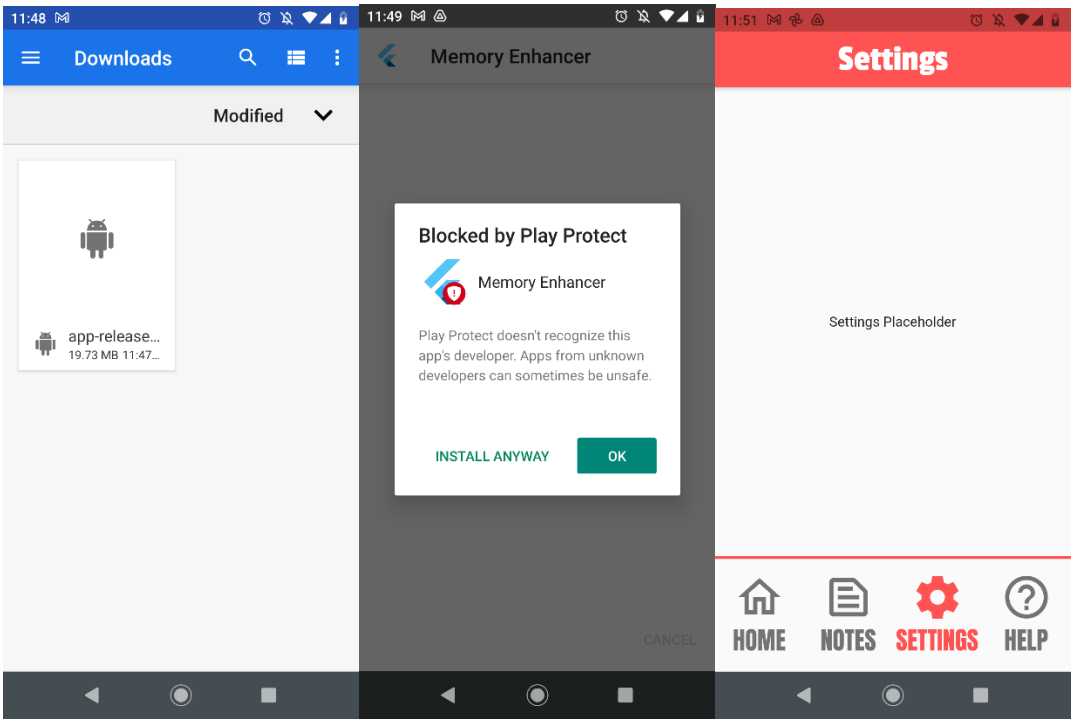



*Figure 23: GitHub Actions - Running APK Artifact on Local Device*

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.

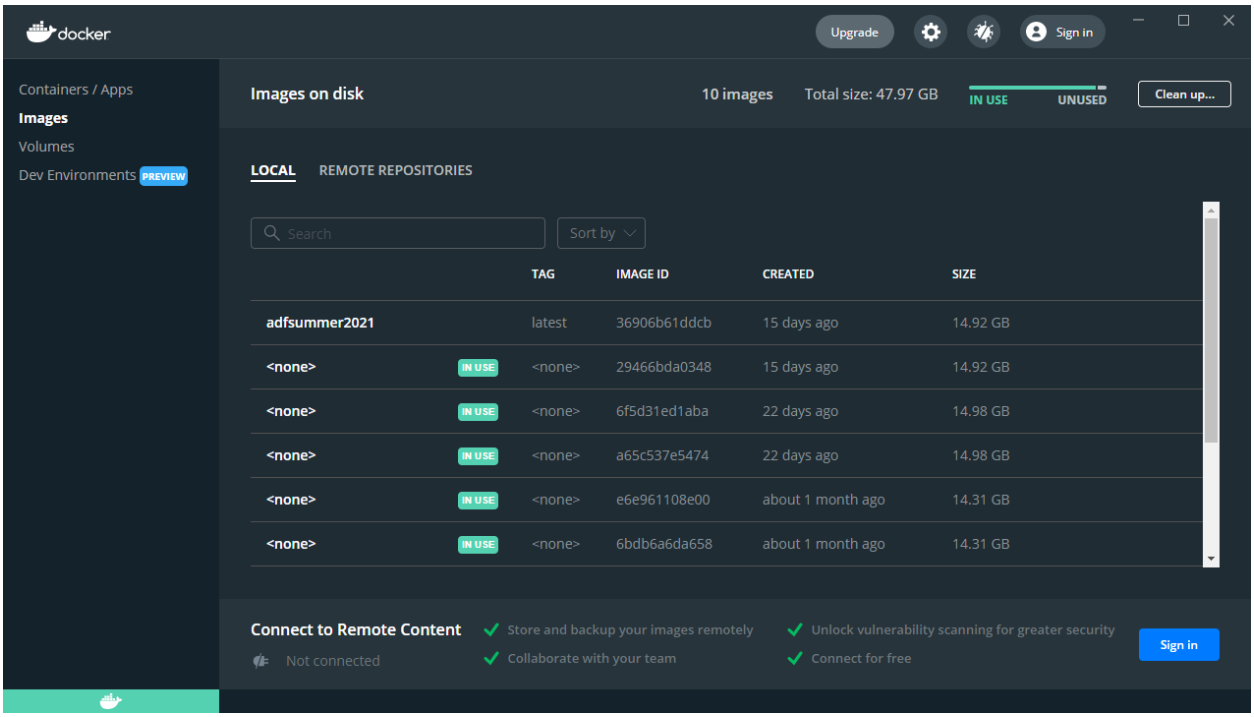



*Figure 24: Docker - Images*

*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\

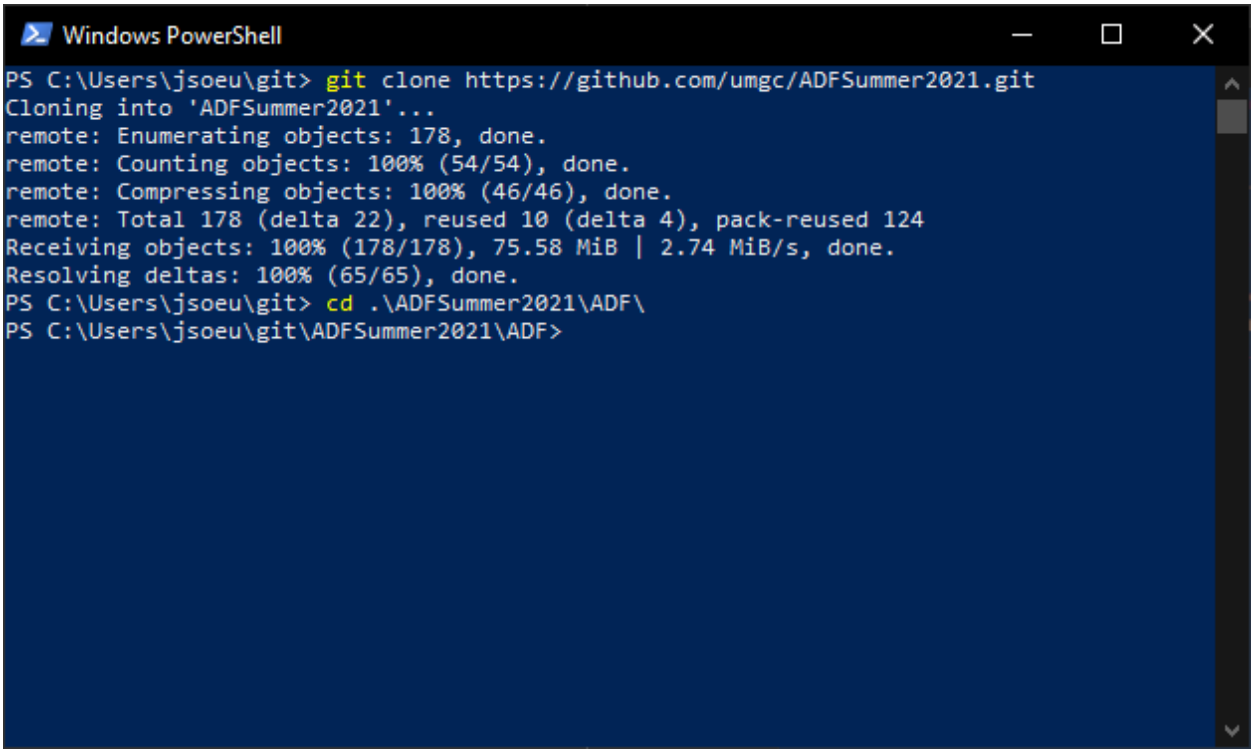



*Figure 25: Docker - GitHub Repository Sync*

***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 mean 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 is will show you Windows (CR LF) in the status bar:*

- *Notepad++*

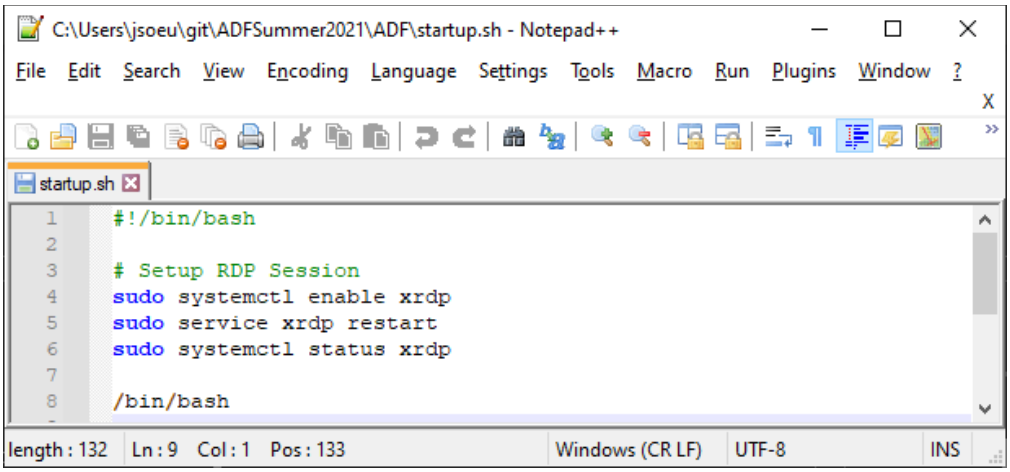



*Figure 26: Docker - Edit startup.sh*

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

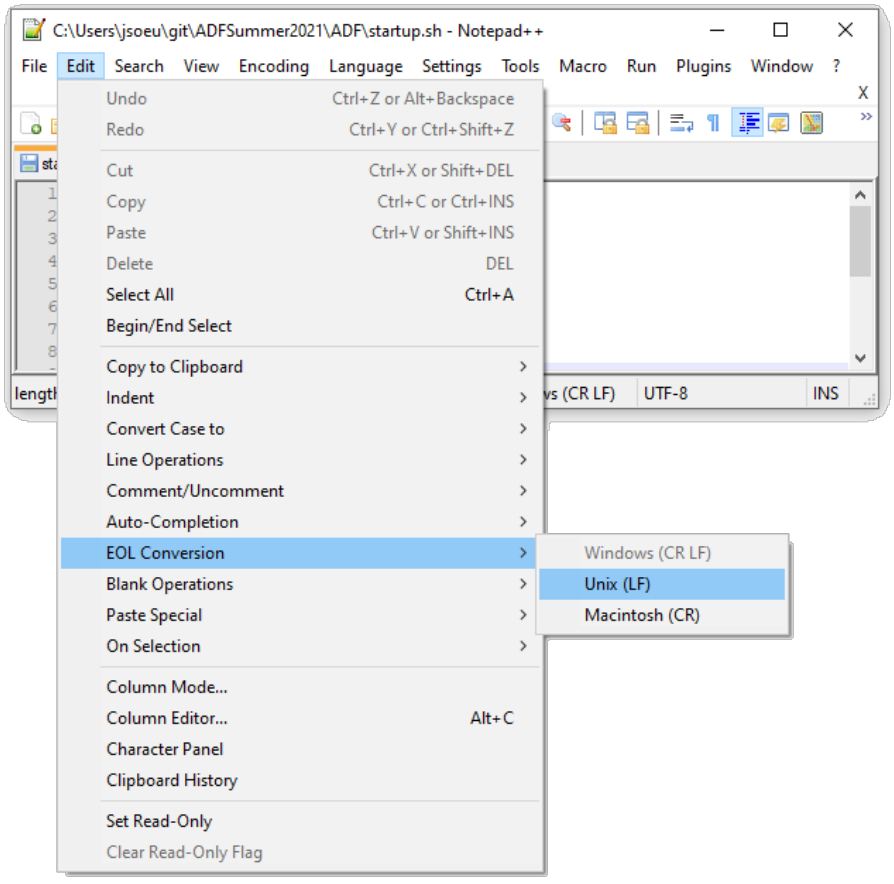
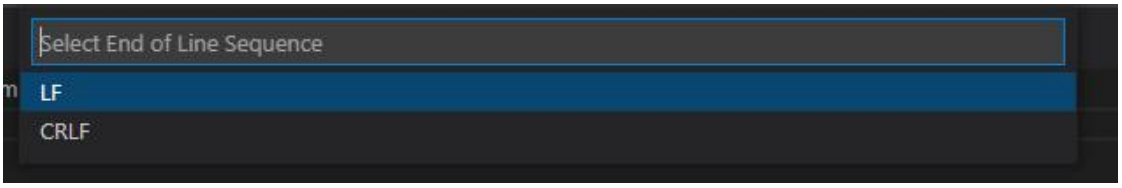



*Figure 27: Docker - Notepad++ EOL Conversion*

- **Visual Studio Code:**

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


Select “LF”




**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:

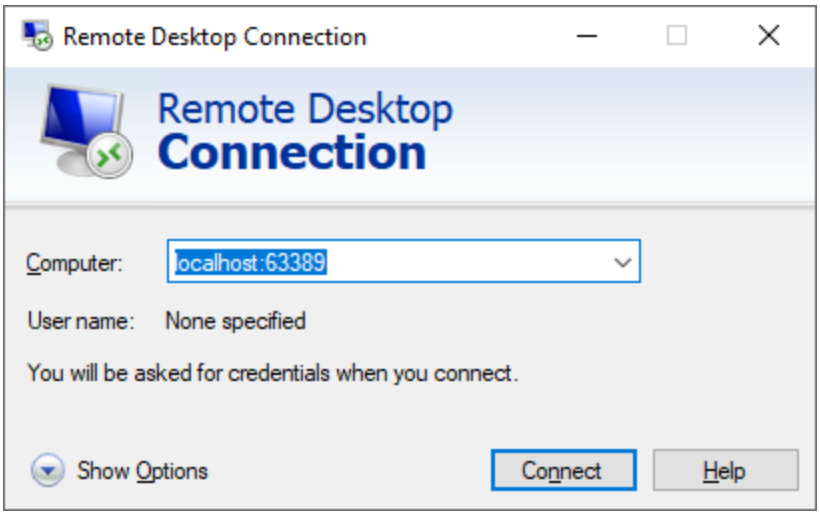



*Figure 28: Docker - Microsoft Remote Desktop Connection Screen*

On Linux, you can

install Remmina.

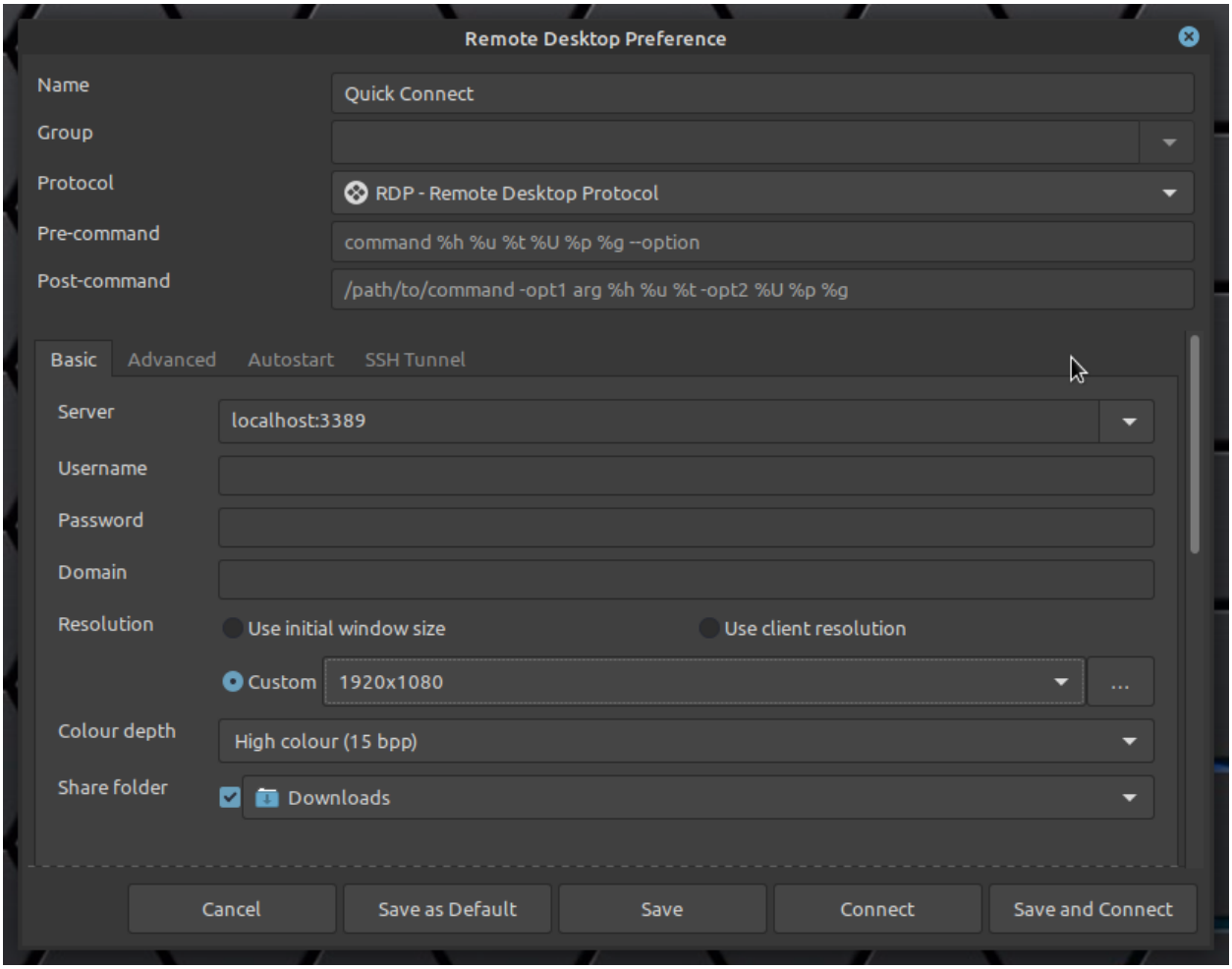



*Figure 29: Docker - Remmina Remote Desktop Preference Screen Part 1*

Make sure to enable sound:

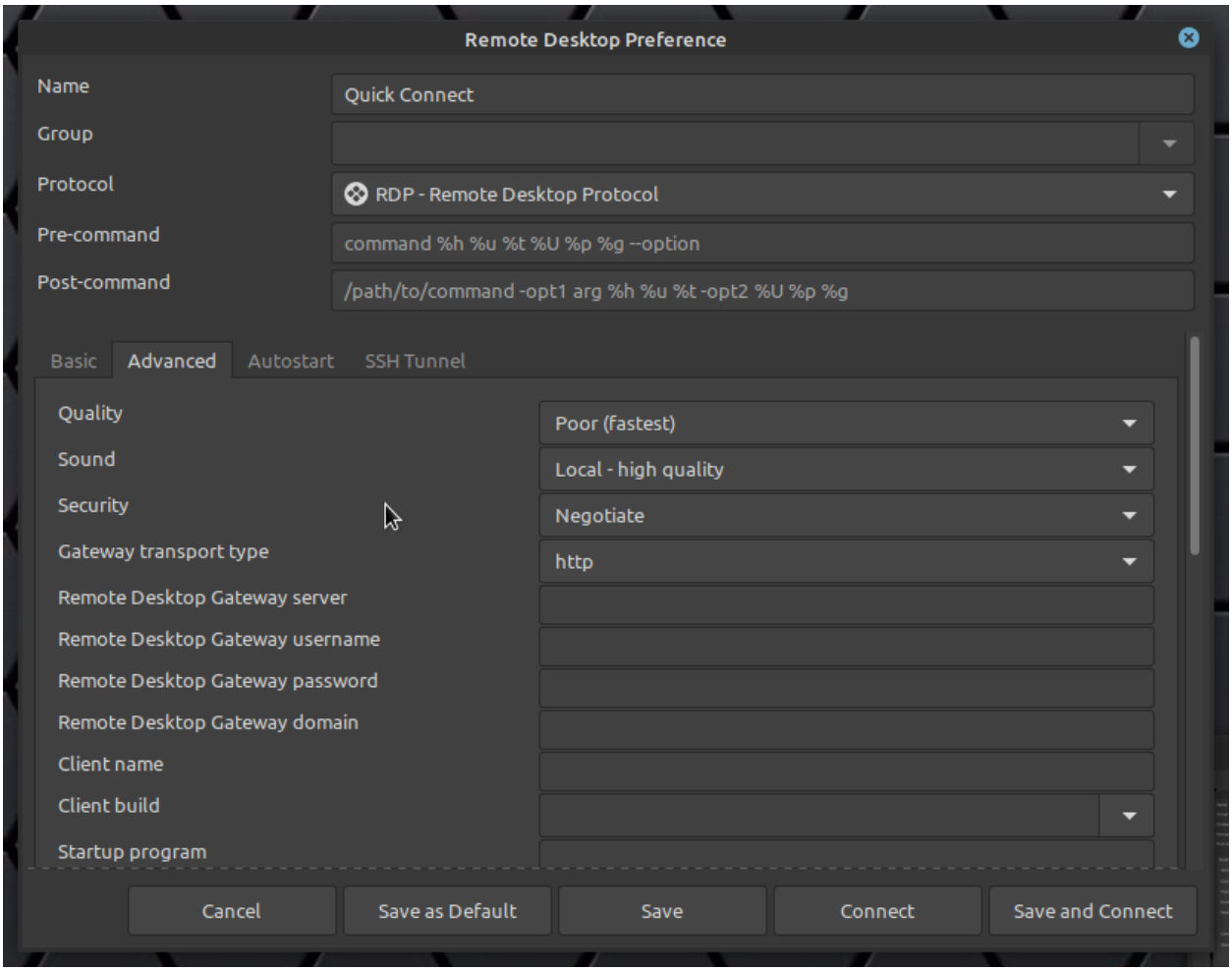



*Figure 30: Docker - Remmina Remote Desktop Preference Screen Part 2*

Next connect and you'll see a login screen.

- Username: developer
- Password: password

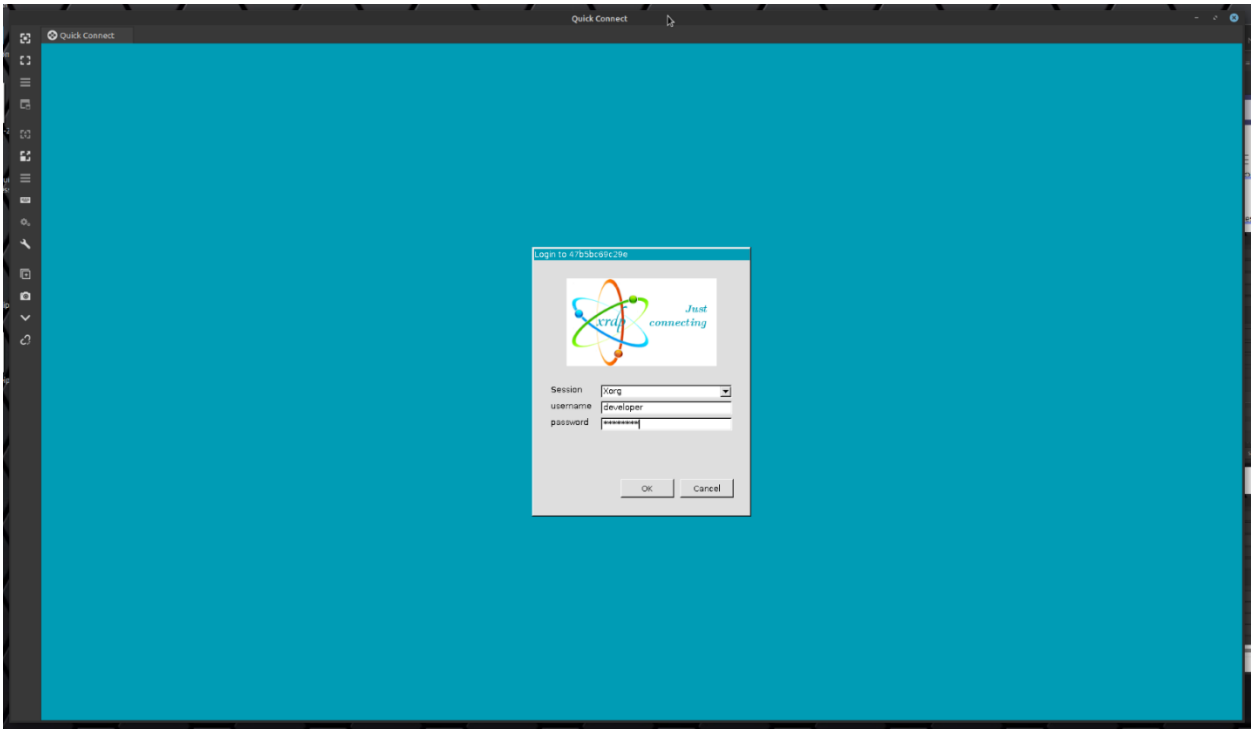



*Figure 31: ADF - Login Screen*

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

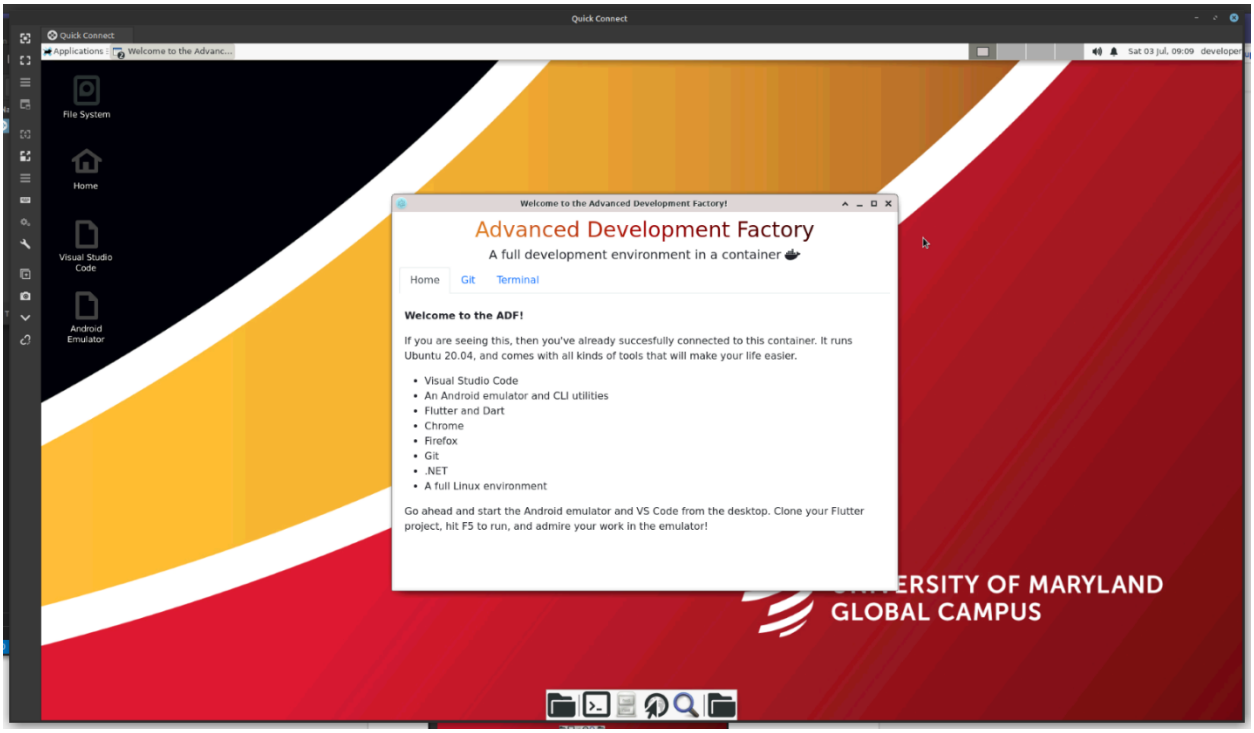



*Figure 32: ADF - Welcome Screen via Remmina on Linux*

Windows:

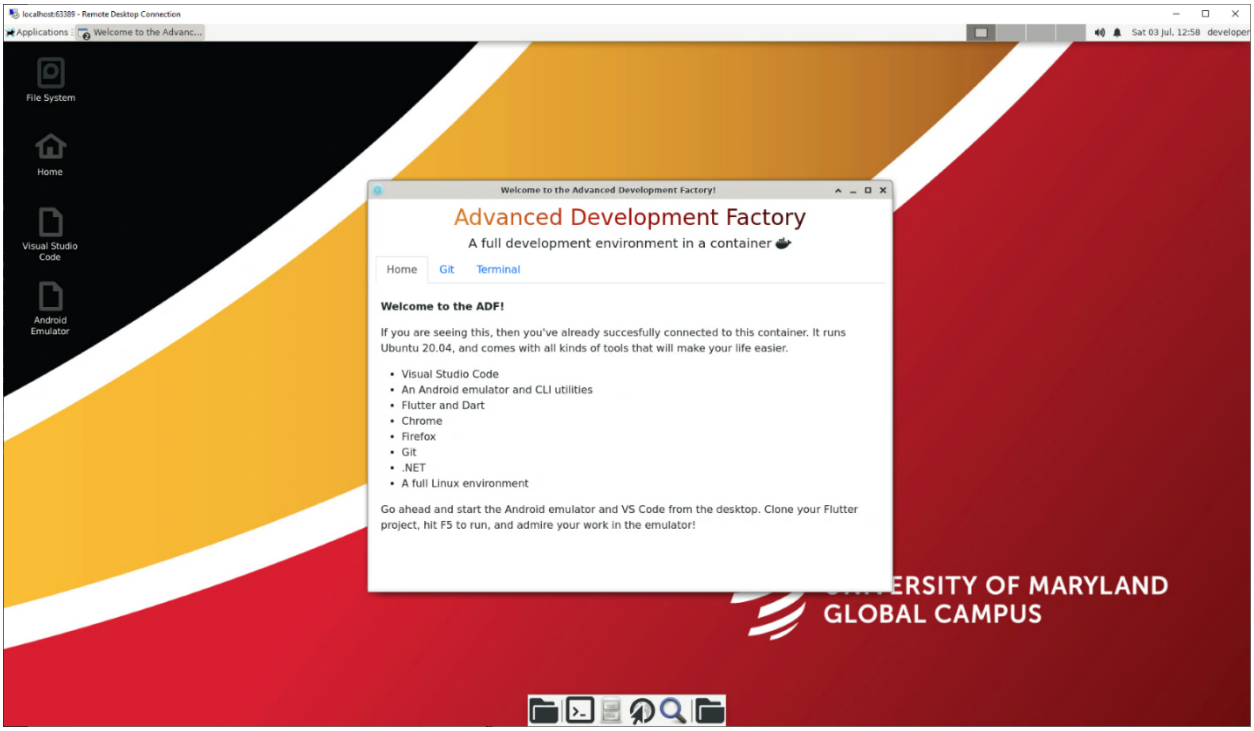



*Figure 33: ADF - Welcome Screen via Remote Desktop Connection on Windows*

**Flutter development in the container**

Start Visual Studio Code from the desktop.

Open a new terminal, and clone the repository:

- git clone https://github.com/umgc/summer2021.charlie.git

Move into the root directory of the app

- cd summer2021.charlie

Start the emulator using (**Note:** The emulator does not work in a container host by Microsoft Windows**) via either of the two following ways**:

1. flutter emulators –launch flutter_emulator
2. from the desktop using the icon.

Run the app using

- flutter run

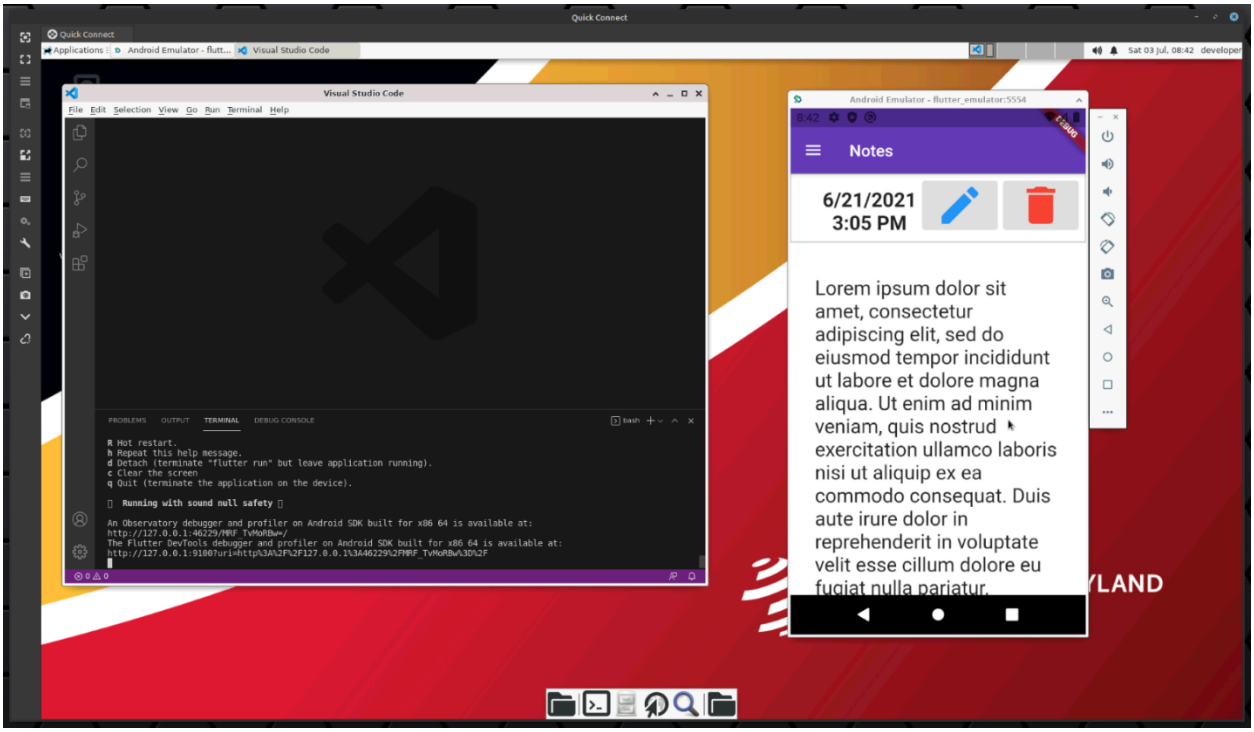



*Figure 34: ADF - VS Code with Attached Emulator*

If you want to make changes, then you first need to tell Git who you are:

- git config --global user.email you@example.com
- git config --global user.name "Your Name"

Then first create a new branch:

- git checkout –b jersoe/buttonfix

Make your changes, and then stash and commit them:

- git add .
- git commit –m “Finally fixed that button”

Then push your changes to GitHub:

- git push -u origin jersoe/buttonfix

**What else is there?**

This Docker image provides you with a full Linux environment. The developer account

has sudo access, so you can install additional software if you want to.

***Note**: sudo access means that you can run any command with root privileges by preceding it with the command sudo (super-user do). Ie: sudo apt-get install vim.*


*Figure 35: ADF - System Info*

**How to install extra software**

First update the deb repositories using

- sudo apt-get update

Next, install the piece of software you need using

- Sudo apt-get install <packagename>

Examples:

*VIM*

sudo apt-get install vim

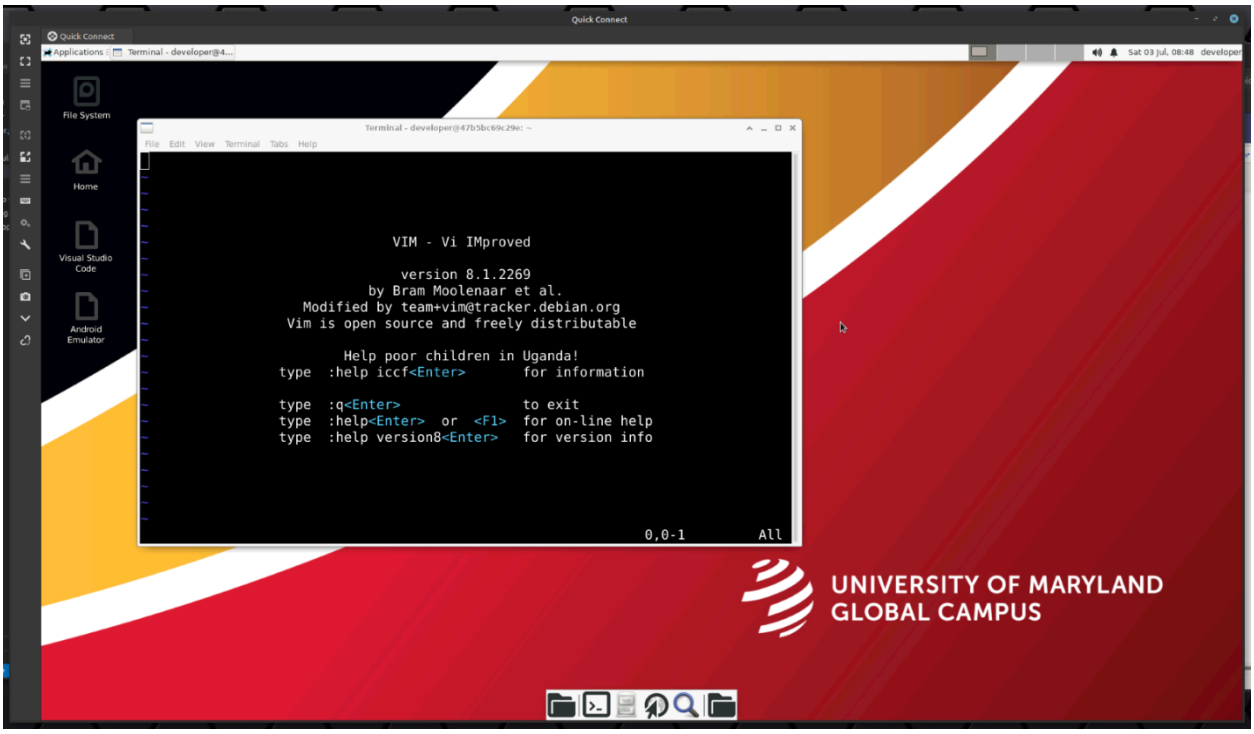



*Figure 36: ADF - Installed VIM*

*Emacs*

- sudo apt-get install emacs

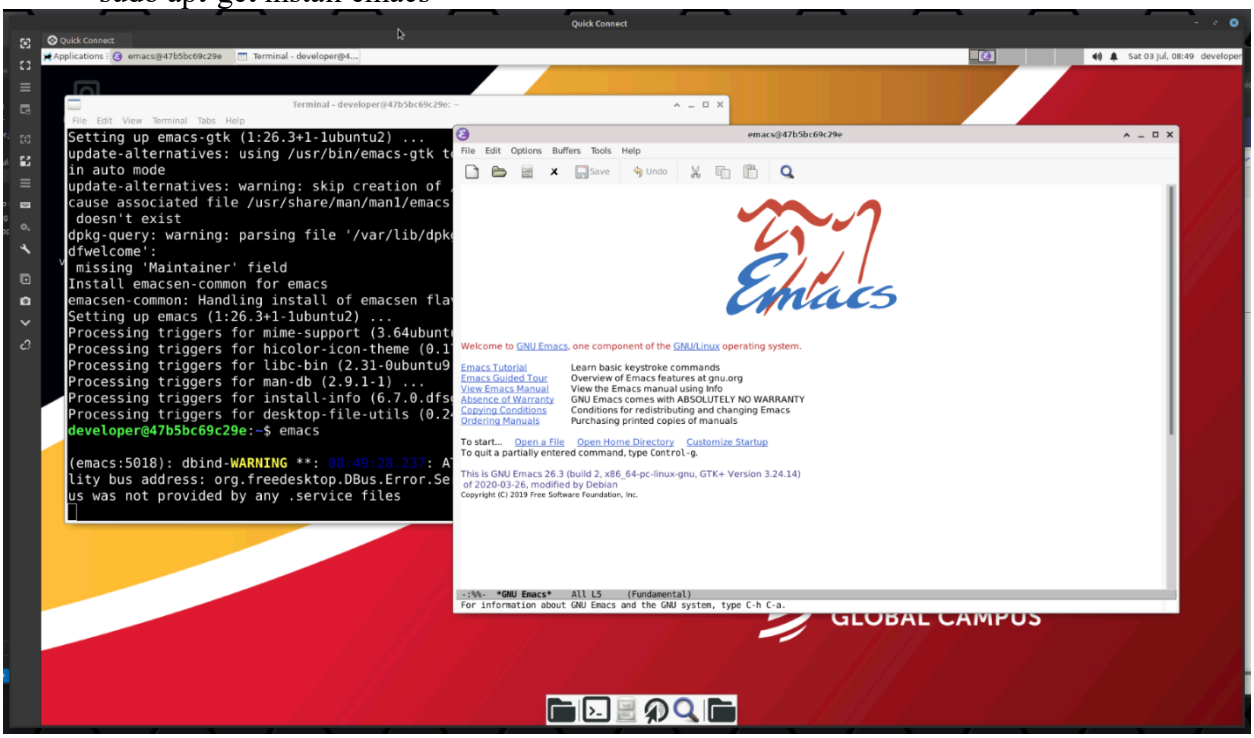



*Figure 37: ADF - Installed Emacs*

*NetBeans*

- sudo apt-get install netbeans

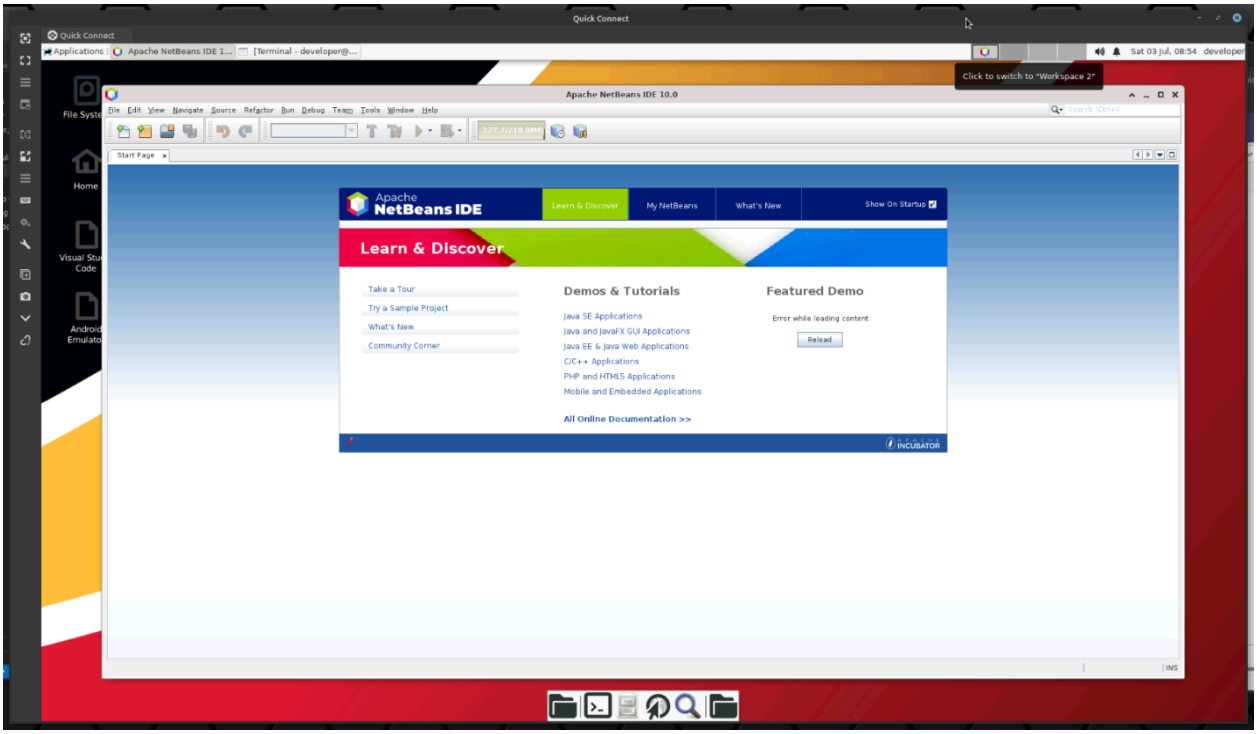



*Figure 38: ADF - Installed NetBeans IDE*

Project idea website

## Introduction

The project idea website allows external parties to request a software development project for the SWEN 670 course. The website has a log-in function where faculty will be able to review, approve, or deny requests. It was developed using the Microsoft Blazor framework, and runs on Microsoft Azure.

## Accessing the website

The website can be accessed on https://umgc-cappms.azurewebsites.net.

## Submitting an idea

The initial page that the user sees contains a form. Here the user can fill out their name, email, project title and description, and upload attachments.

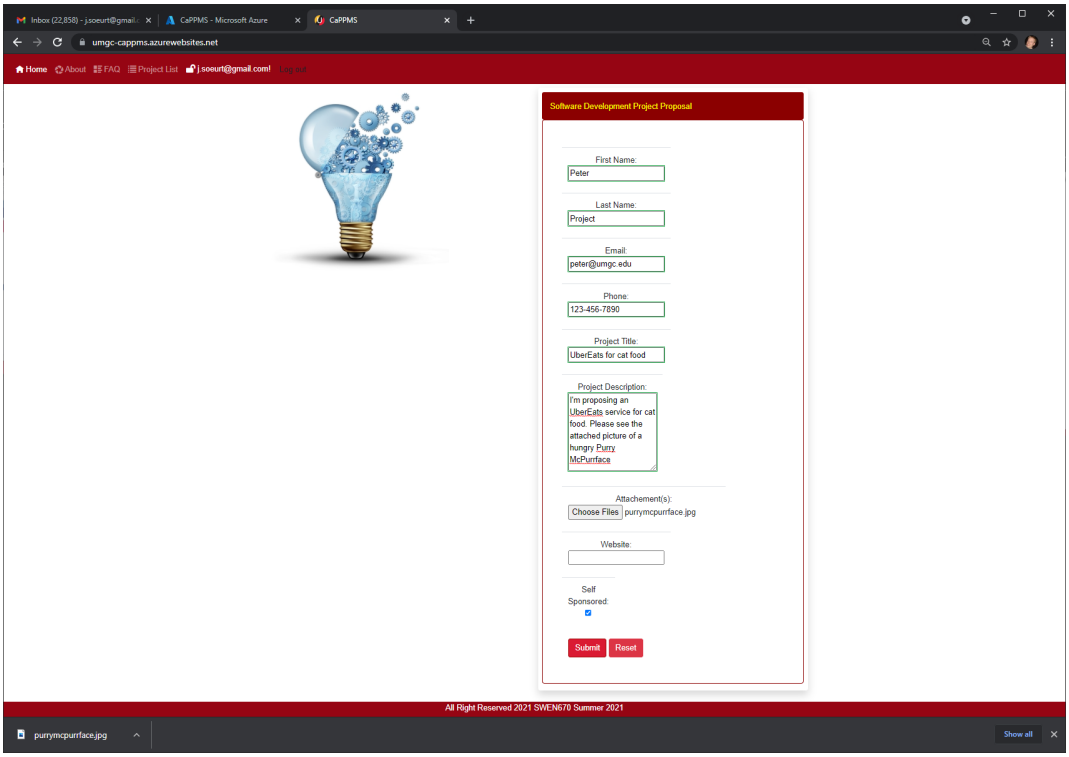



*Figure 39: CaPPMS - Submit Project Idea Page*

If the user unchecks the Self Sponsored checkbox, the form expands to show sponsor information.

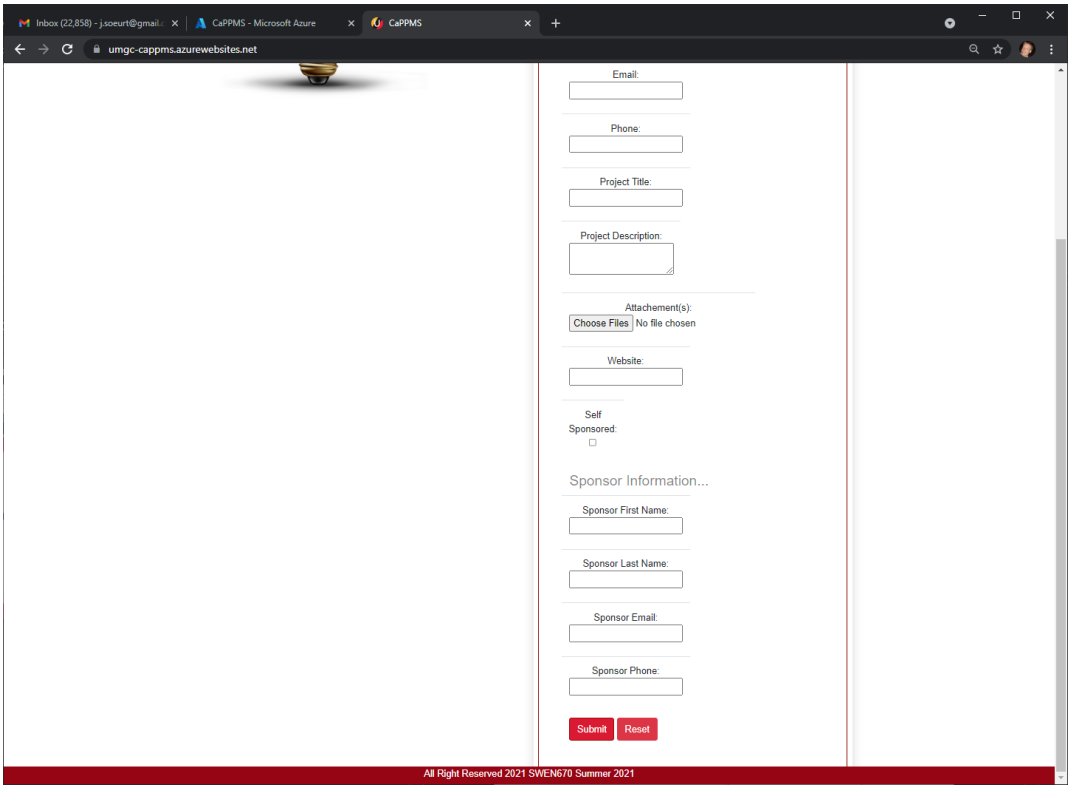



*Figure 40: CaPPMS - Extend Page for Other Than Self Sponsor Information*

If the user is logged in, they can navigate to the Project List:

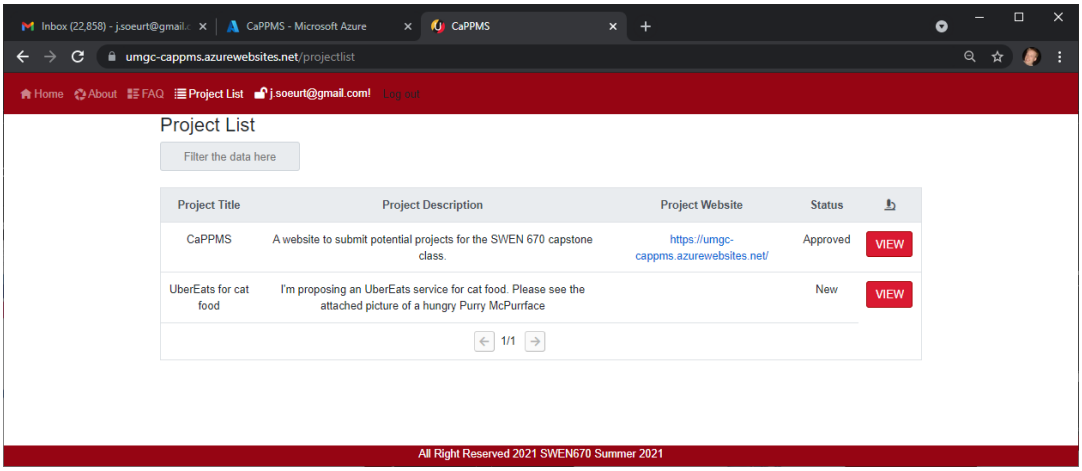



*Figure 41: CaPPMS - Project List Page*

Clicking on View shows all the details.

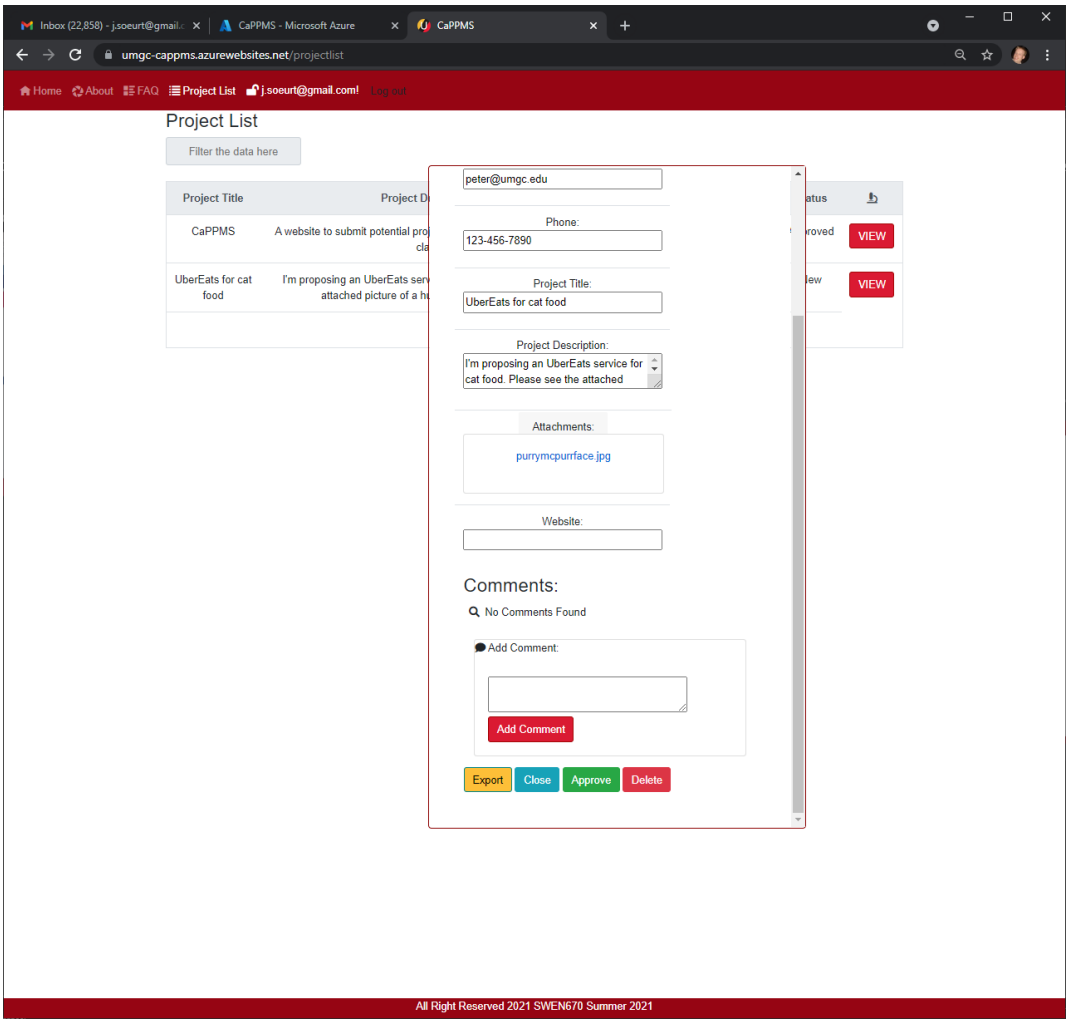



*Figure 42: CaPPMS - Project List Idea View*

Here the user can also choose to export. In that case, a PDF is generated containing the project proposal details.

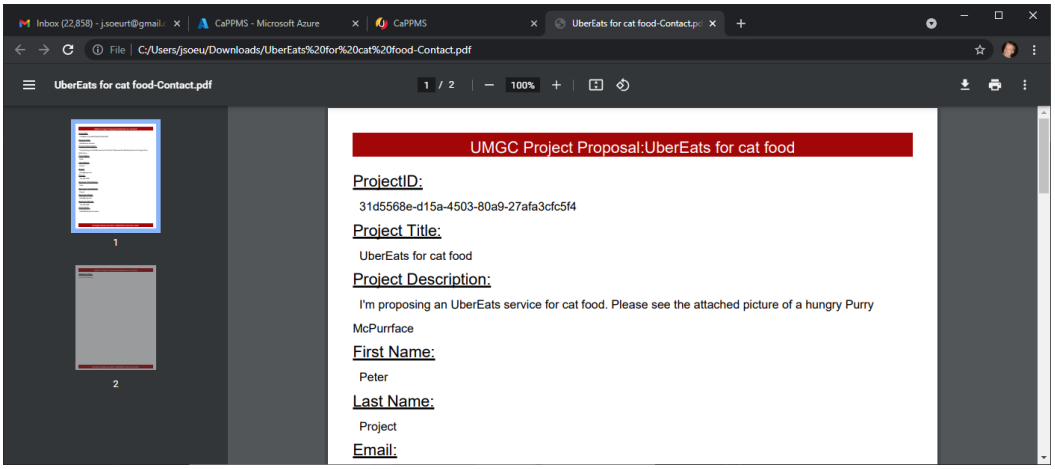



*Figure 43: CaPPMS Project Idea PDF Export*

The logged in user can also leave comments, approve, or delete the proposal.

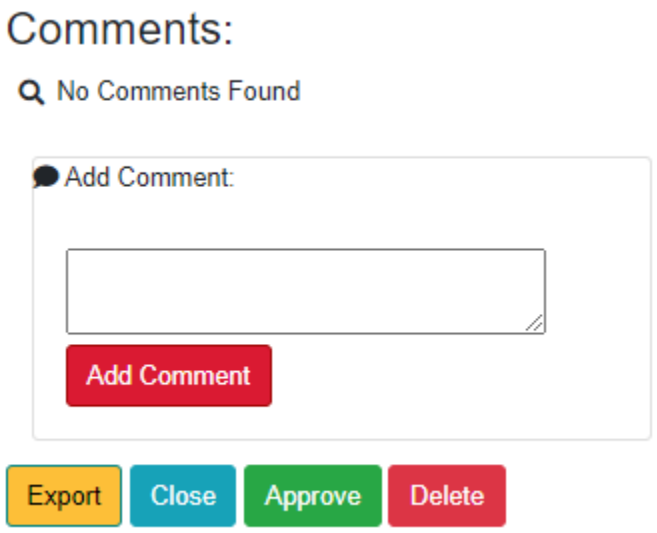



*Figure 44: CaPPMS - Project Idea Comments and Menu Section*