CPSC334
Final Project Reflection
Ian Myers

Converting the previously used project took little work since it was also a python project. I created a new repo and uploaded all my files from my project into the new repo. I then wrote down the list of dependencies I needed to add to my CI pipeline for the project. I then had to adjust my main file that runs everything from a jupyter notebook to a regular python file. I also created a new makefile similar to the one used for this class. From there I started adding the files needed for the debian package and docker image. I created both a debian package and a docker image. I only run the debian package in my CI but the docker image has a make command for it and was run on my local machine. I first started with the deb package by adding the make commands for it. I then copied the code for the debbuild.sh and adjusted it by copying over all the files and folders needed for it. I created a new src directory with all the code and a DEBIAN directory with a control file. I then created a Dockerfile and added the necessary commands for that to run. Minor challenges were encountered from figuring out how to run pip through a container on the CI workflow, figuring out what files are necessary for the debian package without a service and adjusting my make file according to the new necessities. I solved the pip problem with the --break-system-packages tag. This allowed pip to be run even though it was an external system. I realized that the postrm, prerm and postinst were all not necessary since my python script was not a package, therefore, I removed them. I adjusted my makefile to remove the mounting command for docker and test the correct separate testing file. This Linux and DevOps course has taught me how important it is to create structures that allow for repeatability. In the real world, no one really cares if someone has something working on their laptop. What matters is can you take this cool new product and be able to give it to the world. For that to occur, operations need to create workflows and other systems that allow for the product to be rebuilt anywhere. The entire usefulness of a program lies in its ability to be used by other people and other machines. I also learned that as time has gone on, there has been more and more need for abstraction to ensure repeatability. These new forms of abstraction have allowed a lot more people, including me, to create applications that can actually be used by others. Finally, I learned when I should be using the whole hog of my computer and when I can just use a simple GUI.