# Project and testing report

# For Cloud Management System

**Virtual machine creation: in depth timeline & testing.**

The cloud managment system is a GUI enabled application that helps the user with a friendly interface to create a virtual machine with nothing but few clicks away. One of the main challenges was at first finding a suitable virtual machine that has the feasability of being run through a script such as python. Mentioning the coding lanugage now, we've decided to choose python as the source code language for our code for its familiarity and ease of scripting for us.

We've set the users through the application to create a virtual machine in one of 2 ways, a user can select predefined default values or they could simply opt in to custom machine settings, those settings are the CPU cores, Memory size, and finally the disk space.

We've gone with qemu as our platform to run the virtual machine for it's simplicty being run through a simple command line that can easily be implemented through our code. Through that we've faced some issues at first with tests for a basic linux IMG to test and deploy through the code, we believe it should be capable of handling it, however, linux builds often create bottlnecks on low end machines, given that this code was implemented on a low end machine. But we do have faith that the functionality should be alright should the user install a different IMG file to run and use.

During testing for this functionality through our code, we've went with couple versions of linux (due to it's being the most common OS to be virutalized) and some of those distros are antiX and Lubuntu. Currently the software is running on an empty boot for testing purposes and should handle the builds mentioned above or even much higher builds (all depending on the end user machine).

Through our testing for this function we have implemented the minimum and the recommended requirements for running a VM for linux and the system handled it just fine however the bottleneck issues were from the end user machine again given the code is running on a low to meduim end machines during it's testing phase.

**Docker: challenges & functinalities.**

Following on, we've implemented the general and basic features of Docker throughout our project, which comparetivly was easier in implementation and functionality as it has no environment variables as the qemu would require. For that, we've enabled the users to create docker files by specifing a file path and the content. We've also made sure that the program would allow the user to build docker images from a dockerfile as well as showing lthe list of available docker images and docker running containers, all done using the docker commands.

The scrips allows the user to stop a running container using a container ID and it allows them also to search for docker images on the docker hub using docker search functionality that is implemented in the script of this program.

Some challenges that we've faced is locating the directory of the docker file, our understanding of docker implementation itself wasn't as clear, through further research and analysis on the issue we've managed to put together a working program part that helps the user achieve all their computational needs through docker.

**Graphical UI: PySimpleGUI.**

Finally, overall challenges and bumps that we've faced was implementing a proper GUI for this program, as we're using python for this here code, we've concluded after research that pysimplegui was the best library that would aid us in implementing a general, raw, basic GUI using it's amazing and haste functionality for this short deadline. All credits of speeding up the GUI implementation goes to them and for the tutorials provided that helped us import the UI functions to our code.