For this assignment, I decided to make a cinema booking system. The system interacts with 3 VMs (PHP Webserver, Database, and PHPMyAdmin). 2 of our VMs are running Apache to serve web content and our database server is running MySQL. All 3 VMs are running Ubuntu Focal 64bit.

For the first VM this is our main web server, it loads a series of PHP pages which is communicating with the MySQL database VM. The idea of this website is for a customer to come in a selection from a list of currently available movies and able to book a session when that movie is playing. It makes a POST request at the end of the page to the database to add a booking to the booking table. I have coded the PHP page in Adobe Dreamweaver CC 2022 as I have already purchased a Creative Cloud subscription for other projects and work so just trying out this IDE and it seems to be very useful. If I had a better plan for this assignment I could have done some more CSS to improve the layout and design of the page but being that CSS is not being assessed for this assignment I have only done a basic CSS design. To access the web server VM you can access it at http://127.0.0.1:8080

The second VM is running MySQL, this is where all the data for this system is being stored on. This server is only running on the private network so can only be accessed through the web server, phpmyadmin, or ssh into the actual machine if on premise if deployed in the real world.

The third VM is running phpMyAdmin, this VM is intended to be used on premise on the internal network not facing the Internet, used for staff to manage the data in the database. This was the trickiest to setup with vagrant as phpMyAdmin sets up in a similar way as MySQL requiring input from the user while the package is being setup. Another issue is setting up phpMyAdmin to work with a remote database as phpMyAdmin is intended to be installed on the same machine as the database, after many struggles I have finally managed to find a solution to get it installed automatically when vagrant is deployed. To access phpMyAdmin you can access it at http://127.0.0.1:8181
The username and passwords is connected with the MySQL users so you can use the webuser username and password or I have made another account called posuser with the password posaccount

All VMs successfully install their packages and configure everything correctly when you start deploying the vagrant file using the command 'vagrant up'. No manual interaction is required as everything is automated. One issue I have found sometimes running the 'vagrant up' command on my laptop is that sometimes when building the dbserver it might timeout if this happens re provision the machines.

Setup time can vagrant depending on your device and Internet connection. I have been using my main 2 devices to test setup time. On my desktop PC plugged in to Ethernet and having a SSD with an Intel Core i7 and a RTX 2070 it only took around 5 - 10mins setup while my laptop with a HDD and running on WiFi could take up to 30mins.

The main issue with this slow down is with the 'apt-get update' command when downloading packages from Ubuntu's archives. The packages are still set to download from the U.S. servers so thats the main reason for the slowdown. There is also the fact that I have been mainly developing on Windows 10 and Windows 11 and there also seems to be some slow down with using Windows compared to deploying on MacOS or Linux machines. I have not yet had a chance to come and test this on the lab network as the lab network would be significantly faster than my home Internet connection.

One reason to separate these components of this system into 3 separate VMs is security, you can more easily isolate the different components to have different security policies for example you can have only your web server be Internet facing while having your database and phpmyadmin machine local network access only. Another reason if one machine is getting attacked you can shut down that machine and the rest of the system will continue to work as normal while you work on securing that machine that was getting attacked.

One improvement a developer could make is improve could be improve the security of the database and implement some protection from SQL injection attacks, this could be improved SQL queries or improving SQL account security. I have really tested the security of the SQL injection and I could probably imagine that there are some security holes in my PHP scripts. The developer would only have to mainly modified the php scripts in the www/ folder or the vagrantfile or .sh scripts for either the database server or webserver. Depending on the changes if they have only modified php scripts they would have to refresh the pages in their browsers if they have made modifications to the .sh files they would have to redeploy the vagrant machines. They can use the command 'vagrant destroy' to destroy the current machines and then use 'vagrant up' to re provision the machines using the new configurations. A developer could just have to clone the git repository to have everything they need to use and make changes to the vagrant files.

A video playlist is available on YouTube that demonstrates how the vagrant machines work. There is a cut 2 minute version and a full 11min video if more explanation is required - I had to cut a lot of explaining as I like to talk a lot.

COSC349 - Assignment 1 Playlist: https://youtube.com/playlist?list=PLbWdJ74fpcLqOg0fsVYoDPWs10V2I4p0O

Connor Dobson