COSC349 Assignment 1
Jasmine Hindson
8148181

For assignment 1, I have created 3 virtual machines that interact with each other to make a language converter. The first VM that I created is a web server and it displays the current NZ date along with a drop down box where the user can select a language to translate the date into. When the translate button is pressed the NZ date is translated into the language that the user selected and it is shown below. Therefore, the first VM is mainly used to display data from the second and third VM. The second VM is the database server that holds the current language that the user has translated to. The table, dateToConvert, has two columns - *ID* and *lang*. The *ID* column just holds the position in the table, because we are only translating into one language at a time and the *lang* column is the language that has been selected by the user to translate the English date to. The third VM is another web server called translateserver. This web server is used to provide the current NZ date to display on the webserver VM and is also used for all the translations. The webserver VM communicates to the dbserver VM by updating the *lang* column to which ever language was selected by the user. This can be seen as when we select a language from the drop down box and hit translate, in the database table the *lang* column is updated to the new language selected. The webserver VM also communicates with the translateserver VM as when a language is selected, the language value is passed to the translateserver which uses if statements to translate to the language that was chosen by the user.

In the folder www/translate, there is a file called index.php. This is the PHP file used by the translateserver VM. In this file, you can see that I have hardcoded the language conversion because I could not find any open-source APIs that could do the translation for me. I also tried using setLocale() which sets the locale information but this again didn't work for me.

To test the approximate download volumes for the first build compared to the subsequent builds, I closed down my 3 VMs (using vagrant destroy) and then used vagrant up in the terminal to see how long it takes for them to start up after being completely destroyed. For the first time build it takes approximately 6mins and 30 secs. I then used vagrant halt to temporarily stop the VMs from running, and then used vagrant up again to see how long the subsequent builds took. The subsequent build only took 1min and 20secs which is a lot less than the initial build. This is to be expected as the first time starting up the 3VMs would take much longer compared to them already being initialised and just being restarted.

To use the application, you need to download the zip file from GitHub (https://github.com/hinja1997/cosc349). Once the folder has downloaded, you need to navigate to the path of the downloaded folder in terminal. Once you are in the folder in terminal, if you typed 'ls', you should see files like Vagrantfile, main-website.conf, etc. From here, type 'vagrant up' in terminal and you should see all 3VMs starting up. Once they have all started up, open up a web browser and type http://127.0.0.1:8080 (localhost:8080) - this is the webserver VM. You can see on the website that there is the current NZ date and when you select a language from the drop down box and press translate, the translated date is shown below. This is essentially everything to the application and when you want to remove/

delete the application, just got back into terminal into the folder and type ‘vagrant destroy’.

One major useful modification to my application would be to include a Google Translate API to provide more languages to translate to. This would allow the user to select from a much larger range of languages to translate to. I have stated that I had to hard code the languages myself, which can be tedious and leaves room for error. So having a translation API would help a lot with this. I couldn’t use the Goole Translate API as it costs to use and there was a sign up process, which was time consuming.
Another extension I had in mind, was to allow the user to input some text that would be translated into the selected language. Again, this requires something like the Google Translate API since hard coding all the language translations would not be a good idea. To do this idea, you would mainly need to edit the index.php file that the webserver VM uses. In here, you could alter the layout by adding a textfield that gets user input and stores it in the dbserver VM and calls on the translateserver (which has access to the translation API) to do the translation. If any modifications were done to the Vagrantfile, then destroying the VMs and restarting them would be a good idea, since changes have been made to the VMs themselves. If modifications were just done to the PHP files, hitting refresh in the web browser would show the modifications.

Overall, I found this assignment sort of tricky since my language translation idea was not as achievable as I thought. I had to scale my idea down to something that I could implement and get working. I did, however, learn a lot about VMs during this assignment and enjoyed the challenges along the way.