Introduction/Overview:

For this project, I want to develop a Webpage that demonstrates the elements taught in class, which I will address later in my report. Before I go into the specifics of the homepage functionality, I'd want to exhibit and explain the structure of the virtual machines for the project environment. Both of the VMs are using the Ubuntu interface and are connected internally through the adapters. One of the VMs will function as the client (**image 1**), while the other will operate as the server (**image 2**). The client VM will basically upload the webpage and primarily show proof that there is connectivity to the server VM. The server VM will be where most of the main structure of the project will be. For the Server VM, there are two adapters one for internet connectivity and the other will be connected to the client VM. The reason for the internet connectivity is so that we can install Php, Apache Web Server, and MySQL DB Server.

Image 1

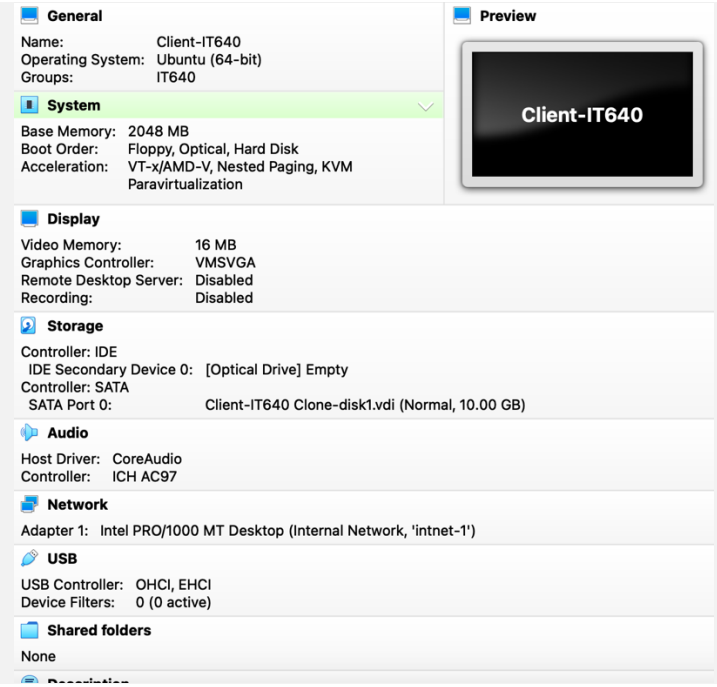
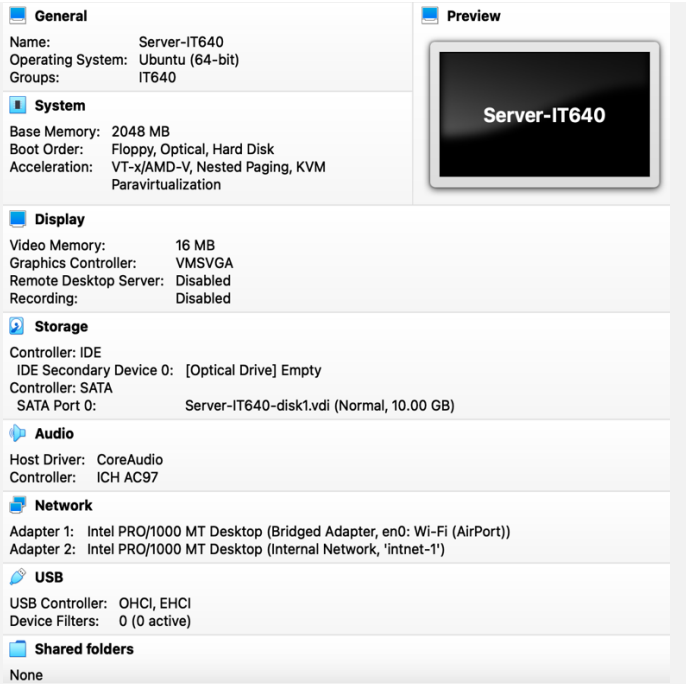


Image 2



Installation of php/Apache Web Server/MySQL:

The project's architect or skeleton is similar to a frontend, backend, queuing system, and database. All of this results in the creation of a webpage. With this in mind, I needed to install services that will be essential in showing the functionality of the project. Ubuntu doesn't automatically have all of the features needed in order for the webpage to function so we have to download each one. The first being PHP. PHP is the language that will be used to link both the front-end, back-end, and the database all together. The command to install php is "*sudo apt-get install php*". We can tell that it's installed because we can see what version of php is on the system (**image 3**). If it isn't installed, terminal will display a "error" message stating that no such application exists. In addition to this operation, we must install an add-on in order for the PHP code to communicate with the MySQL

database. To do so, use the second command, "*sudo apt-get install php-mysql*." With Apache as well we need to install it as well using the command "*sudo apt-get install apache2*". In order to see that it's operational we will use the status command "*systemctl status apache2*"(**image 4**). The final thing that I needed to install is MySQL which is the database that houses the stored values. To install the database, we use the command "*sudo apt-get install mysql-server*". We verify that it's installed using the status command "*systemctl status mysql*" (**image 5**).

Image 3

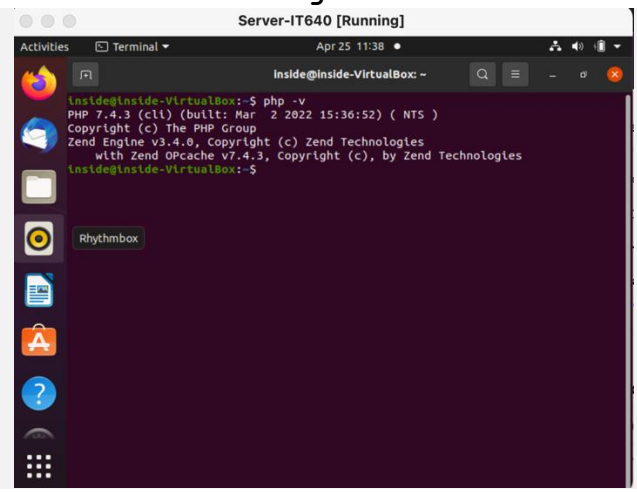


Image 4

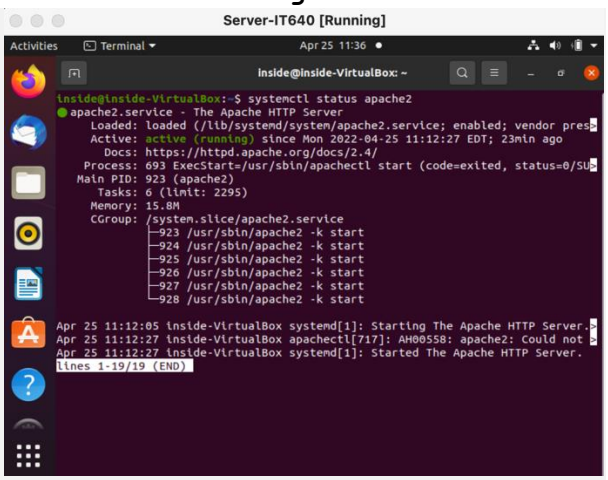
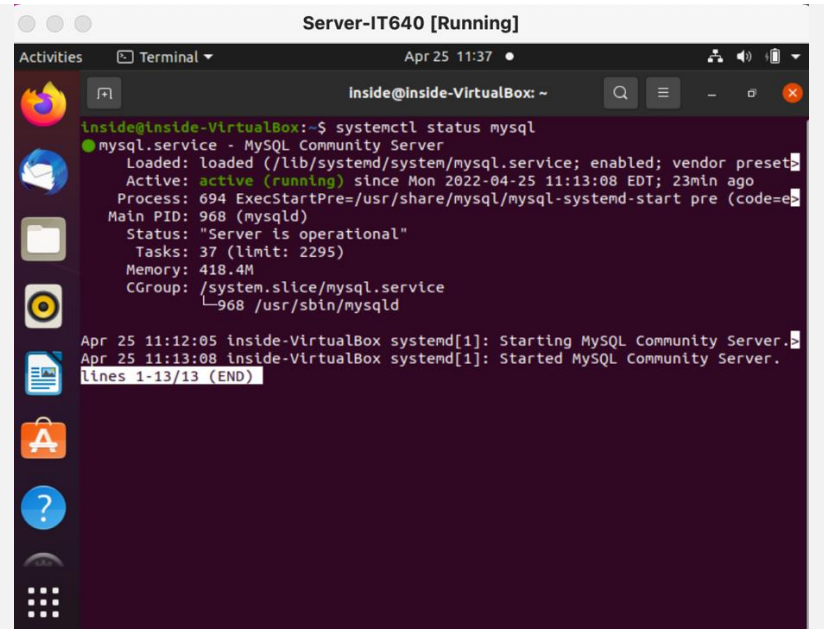


Image 5



MySQL setup:

Before showing the end result of this end-to-end communication between the server and the client, I would like to briefly explain some background set up of the MySQL DB. Aside from building

the table (which will be demonstrated in the following section), various rights and users must be established in order to have some connection and the ability to input queries from the
PHP code. The first command is represented in **image 6** where we create a user and an admin. **Image 7** represent the permission that had to be given in order to make the changes to the table. Once that has been completed you must create the html and the PHP file in the /var/www/html folder which is allowed by the apache2 that we have downloaded.

Image 6

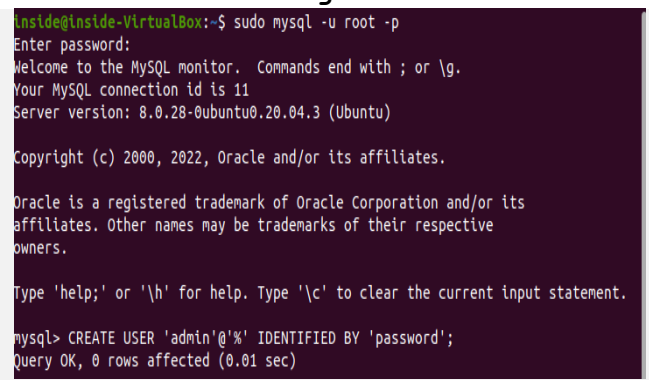
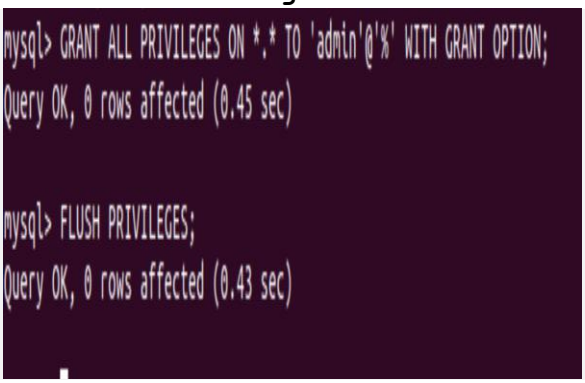


Image 7



Project Results:

With all of these parameters in place, below is the finish product of what we have been setting up for. PHP and HTML scripts were developed for the website (**image 8 & image 9**), as well as before and after images of the client and server screens displaying the updated MySQL DB and the webpage (**image 10 & image 11**). In order to connect to the webpage from the client end, you have to input the IP Address of the server and then the name of the html file. In this case, the IP Address for the server is *192.168.3.1*. Then the name of the html file is *page.html*. You can see these things in the URL portion of **image 10** and **image 11** on the client side.

Image 8

inside@inside-VirtualBox: /var/www/html

GNU nano 4.8 page.html

```
<html>
        <body>
                <form action = "pagephp.php" method="post">
                        Name:<input type = "text" name="name"><br>
                        Email:<input type = "email" name="email"><br>
                        <input type="submit" value="Submit">
                </form>
        </body>
</html>
```

Image 9

inside@inside-VirtualBox: /var/www/html

GNU nano 4.8 pagephp.php

```
<?php

session_start();
$name = $_POST['name'];
$email = $_POST['email'];

$servername =  "127.0.0.1";
$username = "admin";
$password = "password";
$dbname = "home";


//Create connection
$mydb = new mysqli($servername, $username, $password, $dbname);

//Check connection

if ($mydb->errno != 0){
        echo"Failed to Connect to DB" . $mydb->error . PHP_EOL;
        exit(0);
}
        echo "Connection successfully".PHP_EOL;

$sql = "INSERT INTO people (name,email) VALUES ('$name', '$email')";


mysqli_query($mydb, $sql) or die(mysqli_error($mydb));

$conn->close();

?>
```

Image 10

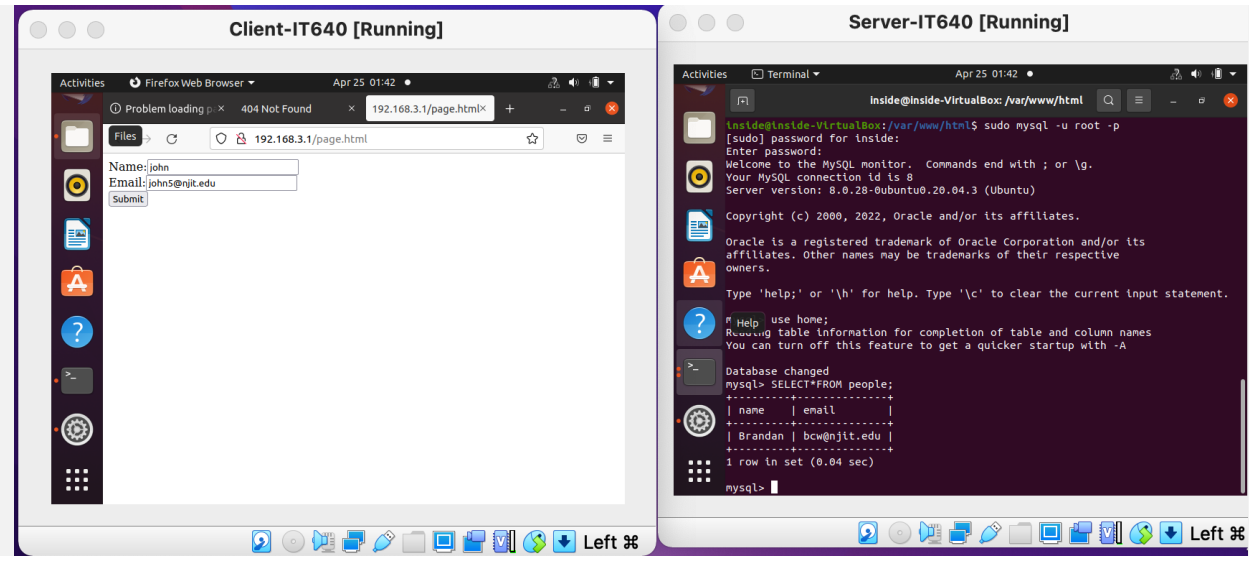
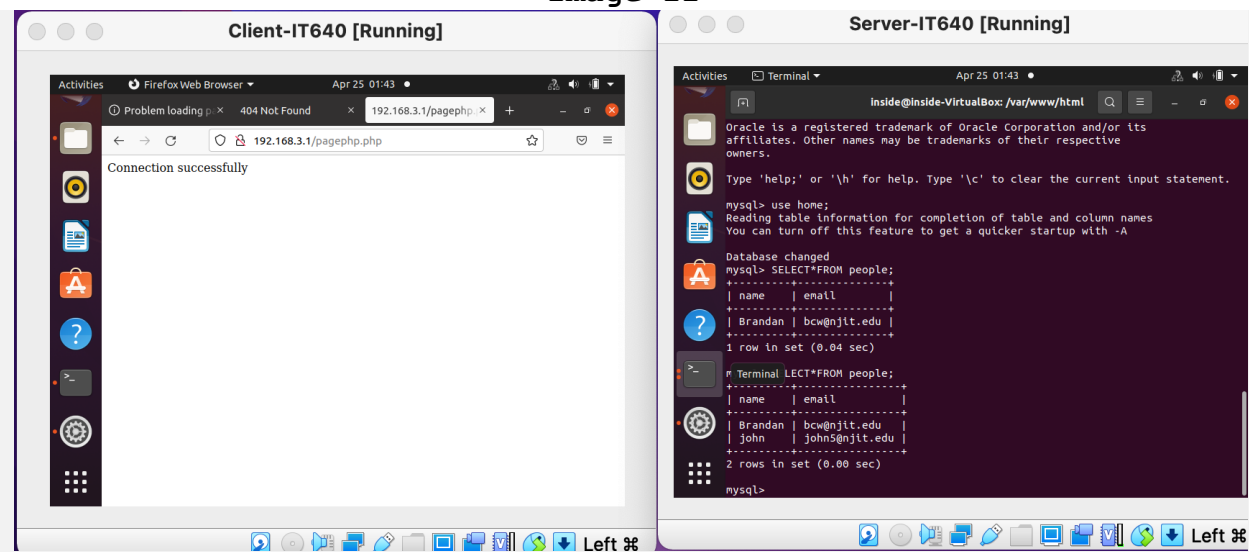


Image 11



Problem / Concerns:

During my process of completing the project, there were some hiccups along the way that took a bit of time. The first thing is making sure that the two VMs were connected. Yes, in the first section of the paper I said how to connection them using the internal network via Virtual Box setup. That's just the first half. Inside each respective VM, you have to set a IP Address for both machines so they can be on the same network and connect to each other. So the IP Address for the server will be *192.168.3.1*. And for the client the IP Address will be *192.168.3.15*. In Ubuntu, here are the settings for the setup (**image 12 & image 13**). Lastly, when you try to save any type of PHP or HTML file on the apache2 /var/www/html directory, you don't have permission to save the file in that directory. This only happens if you download apache2 for the first time. So you have to use the permission command "*sudo chmod o+w /var/www/html*".

Image 12

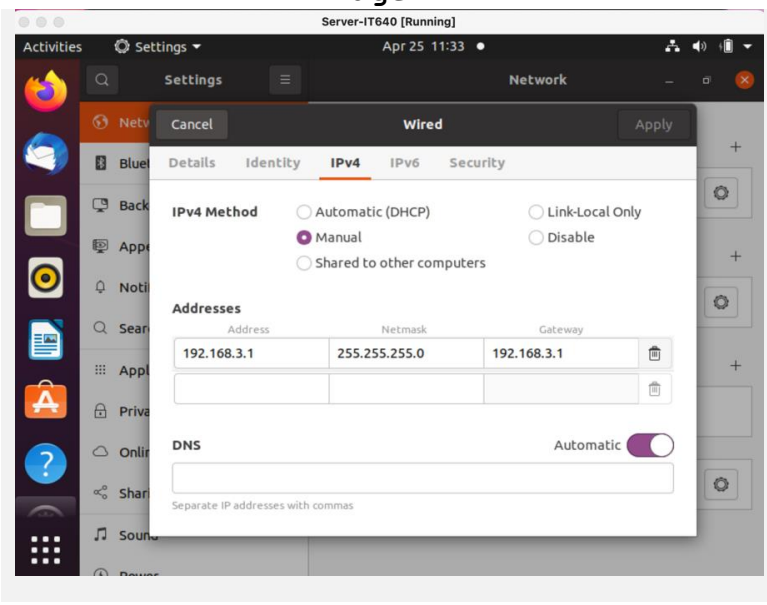


Image 13