VIRTUAL TESTBED

Document: Functional Specification

Student: Adam Kenneth Shibaba

Supervisor: James Egan

23/01/2022

Abstract

The aim of this document is to highlight the functionality this virtual environment will perform. It will display functions and services that will be made available for the parties involved in managing and operating it such as actors. The information provided would also focus on how interaction would exist between users and the application as well as expected result. Finally, I will use this to mention the required deliverables.

Contents

Introduction

Overview

The document is going to focus on the development of a virtual environment. It is my hope to see that modules across Year 3 and 4 cybercrime and IT Security department will be incorporated into virtual machines to facilitate learning for students. The environment will contain efficient virtualization software and automation such as Docker and ansible. Furthermore, the project would include the use of a popular cloud platform to ensure convenient access, user experience, and security

Functionality Specification Scope

In this project, I will present the implementation of a virtualized system that will contain several technologies to help store applications required for 3rd or 4th year cybercrime and IT Security modules. It assists students with access to several tools necessary for practical operations to help them achieve relevant IT skills. For example, some models such as forensics, Reverse engineering, networking, and Software Architecture utilize software to accomplish several tasks. Through this project, software can be stationed in a containerized environment or server, that students can access and explore. The use of automation software will create servers with containerized virtual machines and API to configure and deploy all tools. The system will showcase the efficiency of a type of virtualization that utilize resources of the Operating System rather than hardware.

Technologies

- Ansible automation software
- Apache, httpd, ssh
- Docker container modules
- Linux operating system
- Linode
- LAN

Application Users

Administrators

The importance of administrators is crucial because they ensure privileges are deployed to users and security is maintained. Students operating VMs can unknowingly disrupt the network that virtual machines sit on, thereby disturbing their performance. Administrators would help prevent this because of their constant monitoring and supervising. Also, they put policies in place to help protect the system. This way, no resource is wasted.

Students

Students would have regular interactions with virtual machines since they are limited with the privileges they have; this means they would not be able to perform modifications on VMs. Only administrators/tutors can enable this. In scenarios where the application they are working with requires ad-on, plugins or extensions, then administrator would grant them permission to include such tools. Students would not be able to view other VMs apart from their created VM

Tutors

Tutors have higher permission over students. Such permissions allow them to assess practical work being done on students' VM, from a primary host machine. This way tutors can assist students with difficulties. Tutors would have permission to include ad-on, extensions or plug-ins required for tutorials, view, delete, assess and modify VMs.

Use Case and Scenario

Login Primary Actor

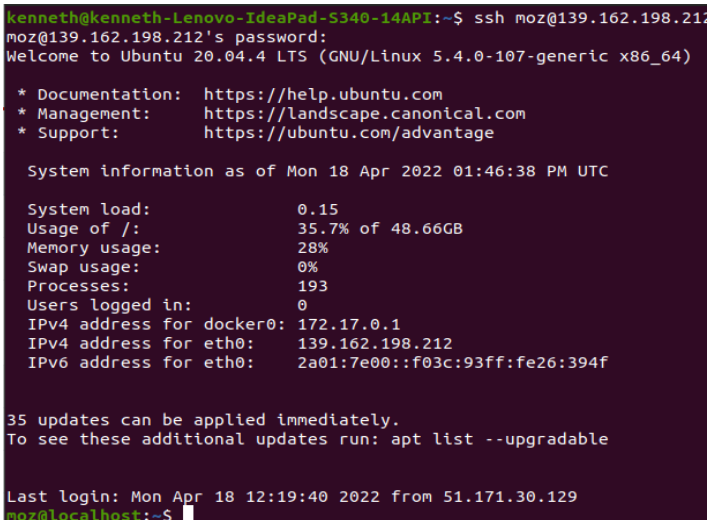
Administrator:

Preconditions:

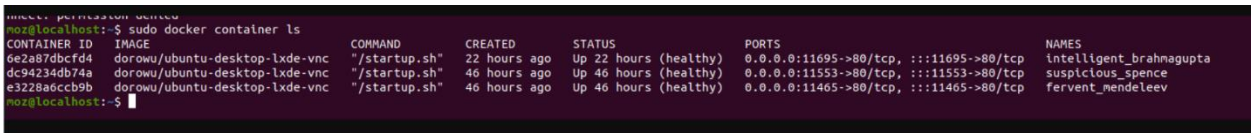
Administrator has created server with docker containers, linode, and required services e.g webserver, https, mysql, php, and webapplications

Main Scenario

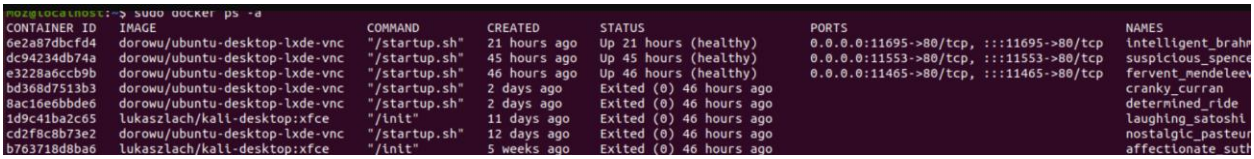
1. Administrator remote login



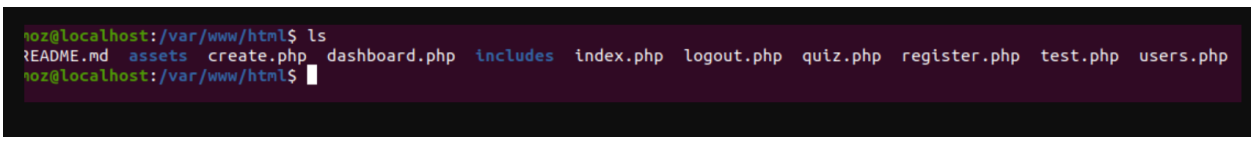
2. View docker and docker images



3. View running containers

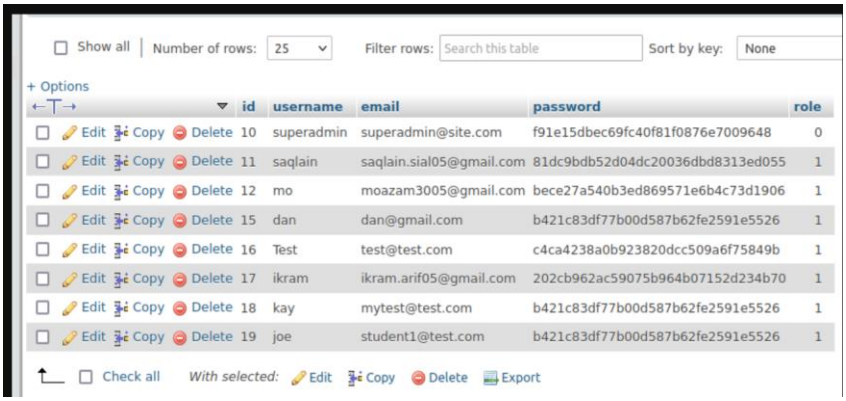


4. Source codes

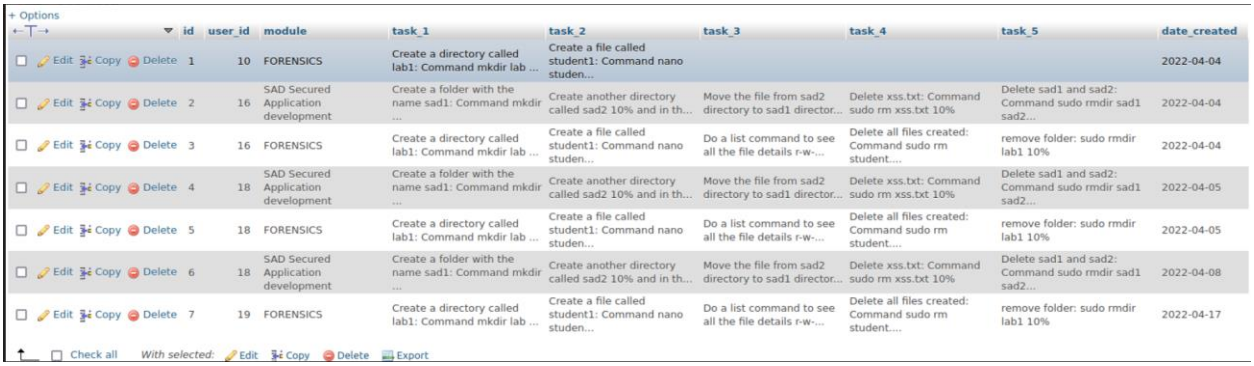


5. Database login

6. User credential table



7. Quiz table



Result

Successful

Administrator is able to

- Login remotely
- Deploy docker in server
- View docker and docker images
- View running containers
- View Source codes
- View or modify all tables in the database

Login Primary Actor

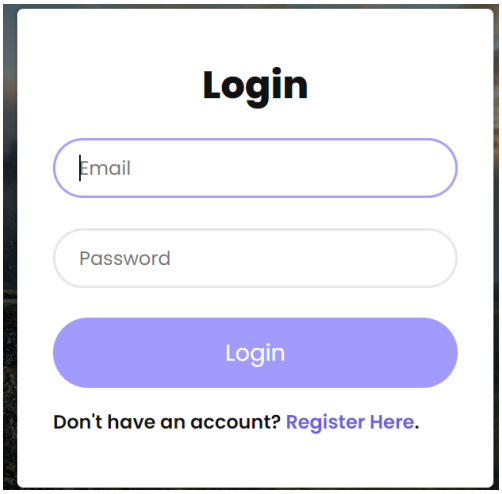
Tutor:

Preconditions:

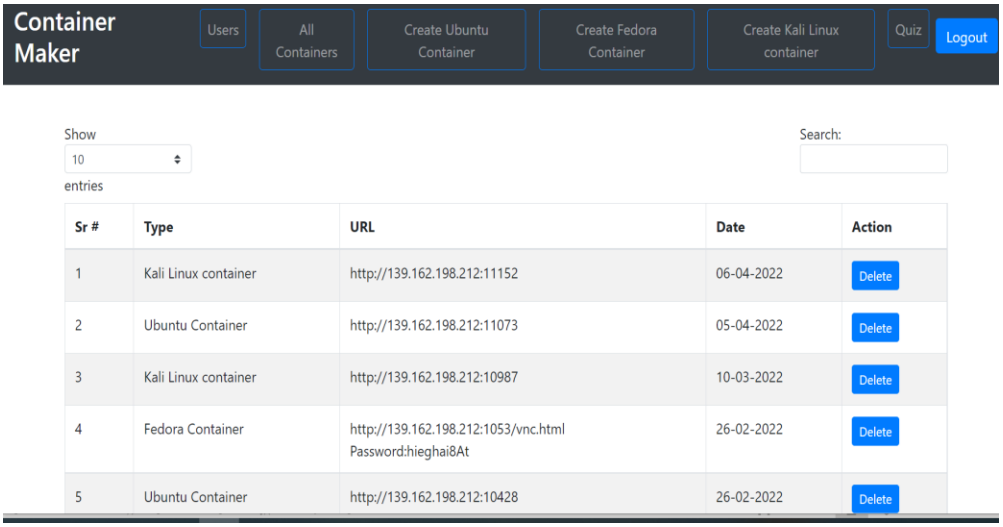
Tutor has been provided an account and credentials to access the server

Main Scenario

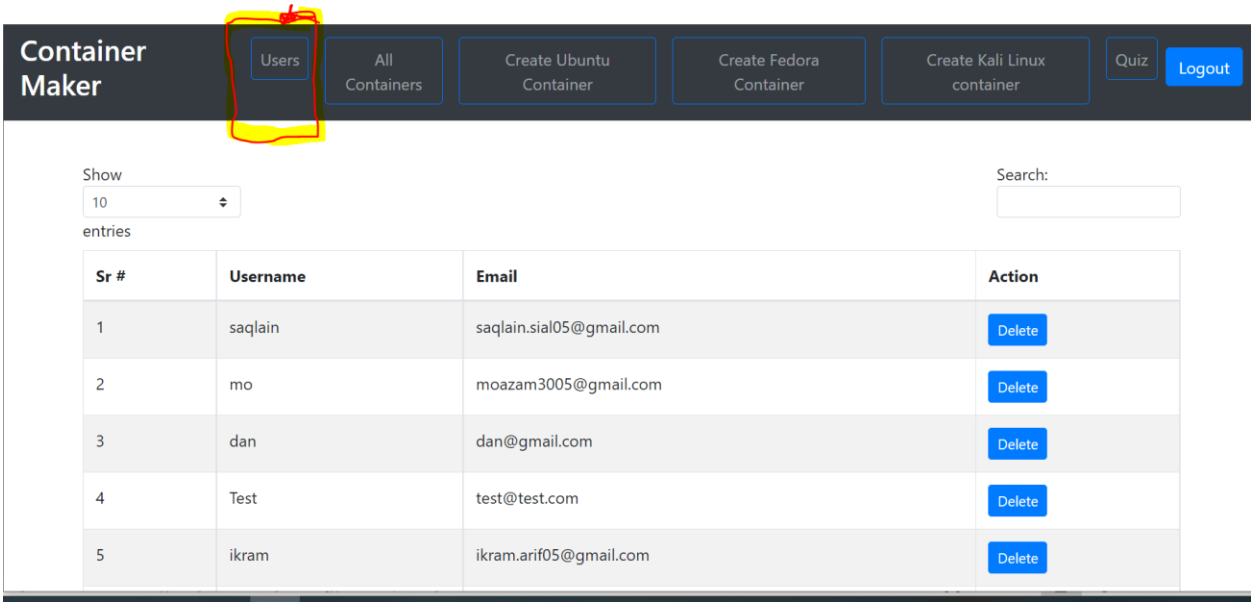
1. Tutor will launch web application
2. Login page will load for tutor



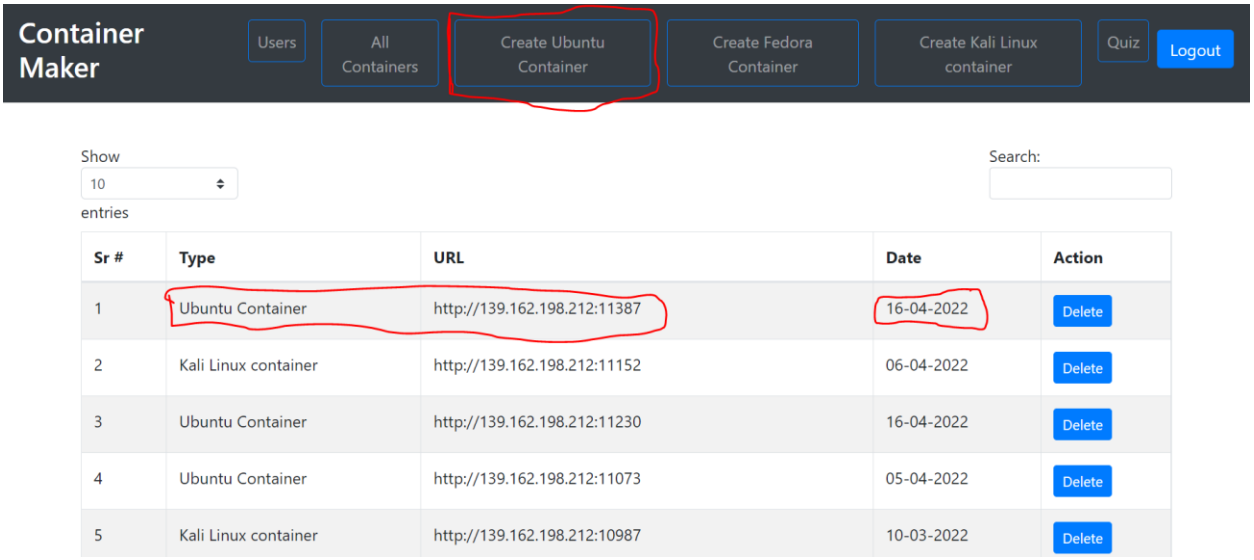
3. Enters login credential. Username and password
4. Login details are then validated by database
5. If validation is successful, then tutor will be granted access.
6. Access applications. View Containers created



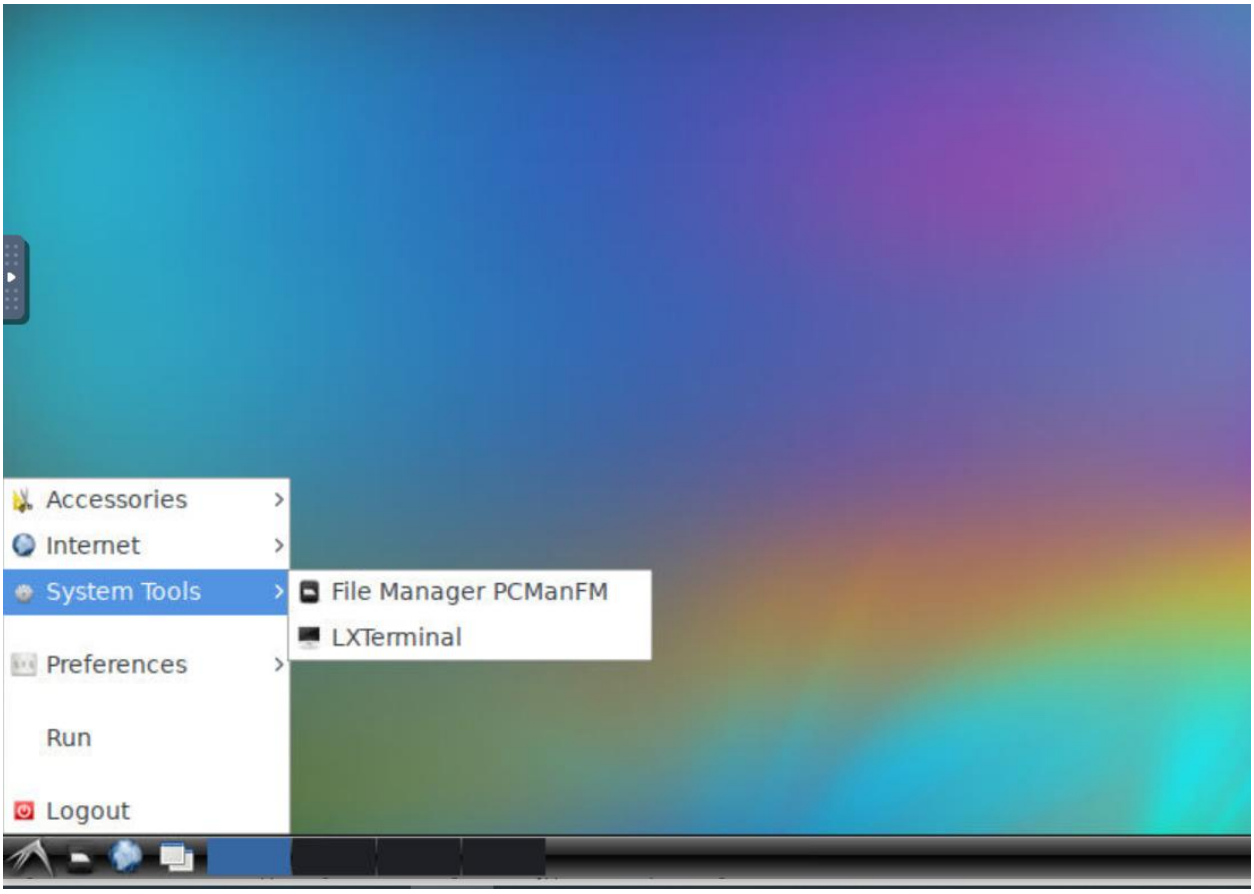
7. View Users and their containers



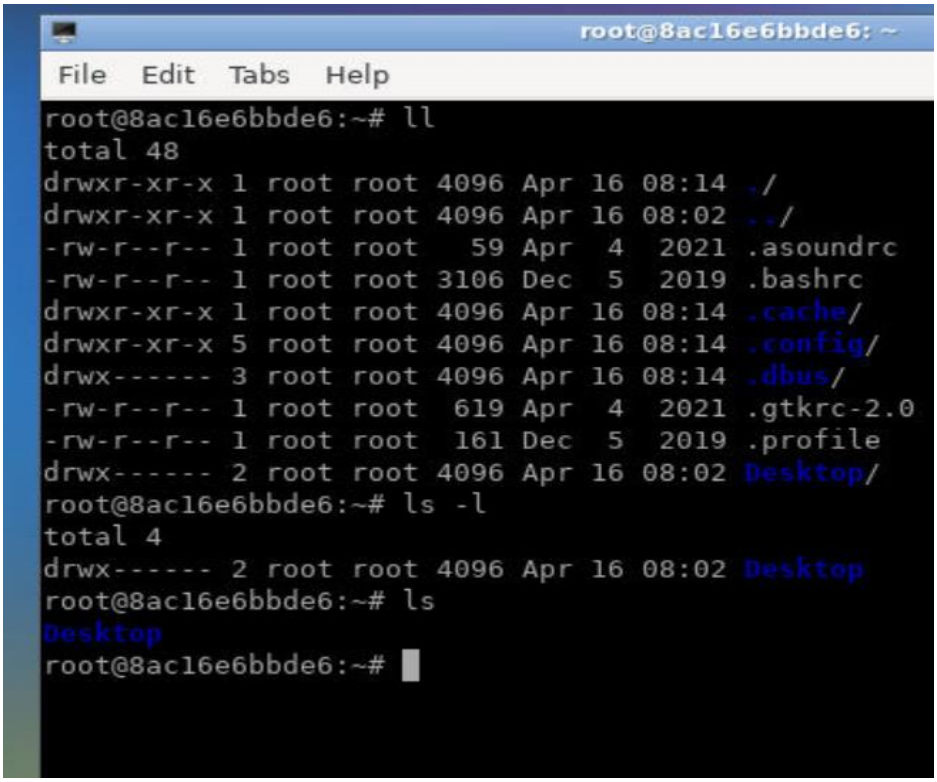
8. Create VM



9. Access students VM or Created VM

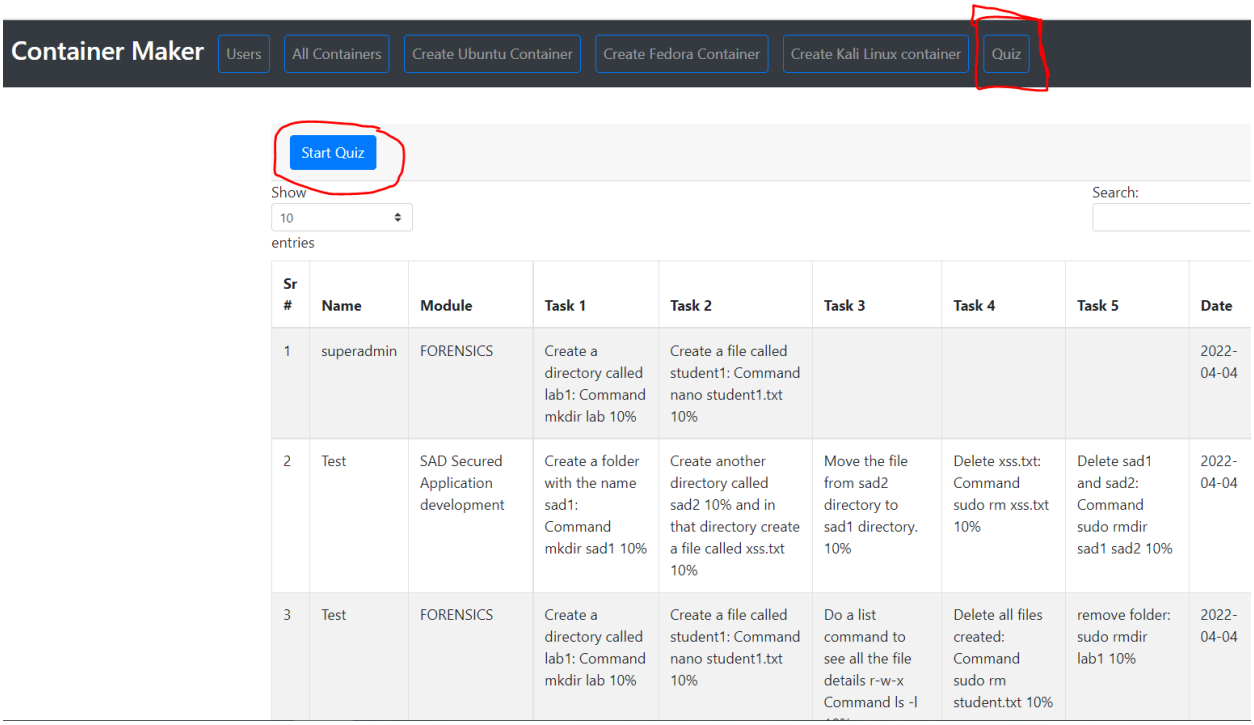


10. Terminal



11. Check for completed tasks

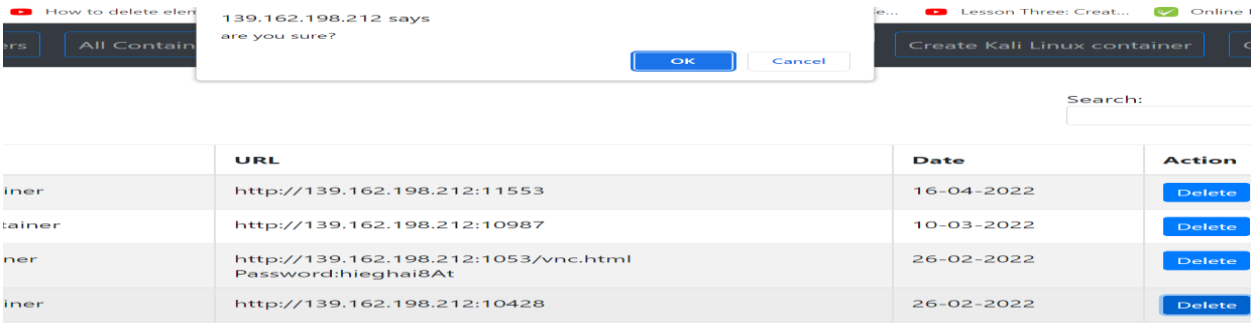
12. View Tasks/Quiz



Sr #	Name	Module	Task 1	Task 2	Task 3	Task 4	Task 5	Date
1	superadmin	FORENSICS	Create a directory called lab1: Command mkdir lab 10%	Create a file called student1: Command nano student1.txt 10%				2022-04-04
2	Test	SAD Secured Application development	Create a folder with the name sad1: Command mkdir sad1 10%	Create another directory called sad2 10% and in that directory create a file called xss.txt 10%	Move the file from sad2 directory to sad1 directory. 10%	Delete xss.txt: Command sudo rm xss.txt 10%	Delete sad1 and sad2: Command sudo rmdir sad1 sad2 10%	2022-04-04
3	Test	FORENSICS	Create a directory called lab1: Command mkdir lab 10%	Create a file called student1: Command nano student1.txt 10%	Do a list command to see all the file details r-w-x Command ls -l	Delete all files created: Command sudo rm student.txt 10%	remove folder: sudo rmdir lab1 10%	2022-04-04

13. Delete VMs

1	Ubuntu Container	http://139.162.198.212:11553	16-04-2022	Delete
2	Kali Linux container	http://139.162.198.212:10987	10-03-2022	Delete
3	Fedora Container	http://139.162.198.212:1053/vnc.html Password:hieghai8At	26-02-2022	Delete
4	Ubuntu Container	http://139.162.198.212:10428	26-02-2022	Delete



entries

Sr #	Type	URL	Date	Action
1	Ubuntu Container	http://139.162.198.212:11553	16-04-2022	Delete
2	Kali Linux container	http://139.162.198.212:10987	10-03-2022	Delete
3	Fedora Container	http://139.162.198.212:1053/vnc.html Password:hieghai8At	26-02-2022	Delete

Showing 1 to 3 of 3 entries

Previous 1 Next

Result

Successful

Tutor is able to

- View All users and All containers
- Create VM
- Access students VM
- Enter terminal
- Check for completed tasks
- View tasks/Quiz
- Delete VMs
- Logout

Login Primary Actor

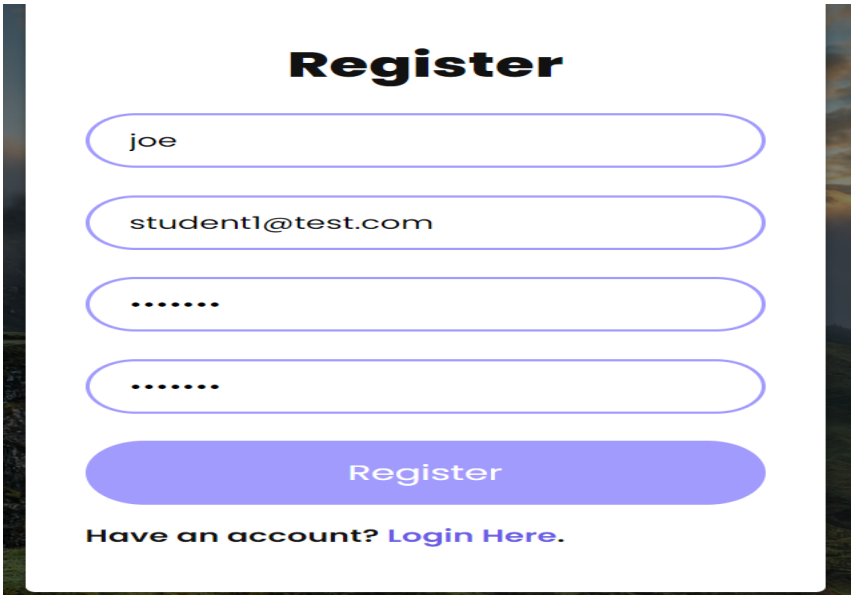
Student:

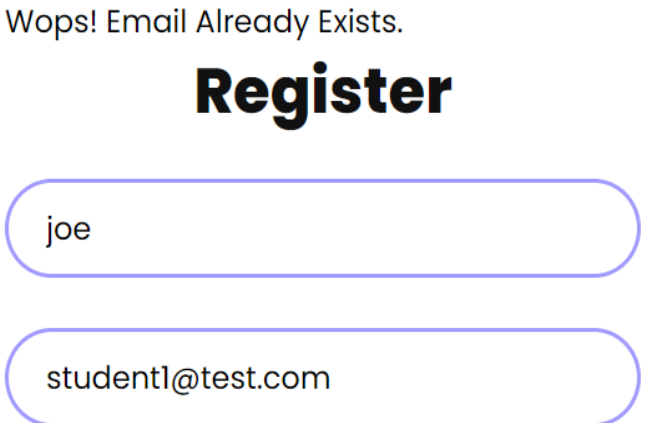
Preconditions:

Student will have to register before he or she can sign in

Main Scenario

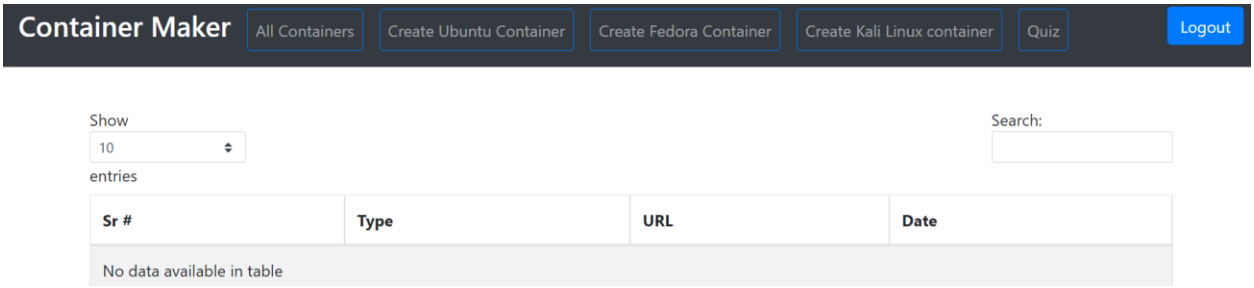
1. Student will launch web application
2. Login page will load for student. Student would register



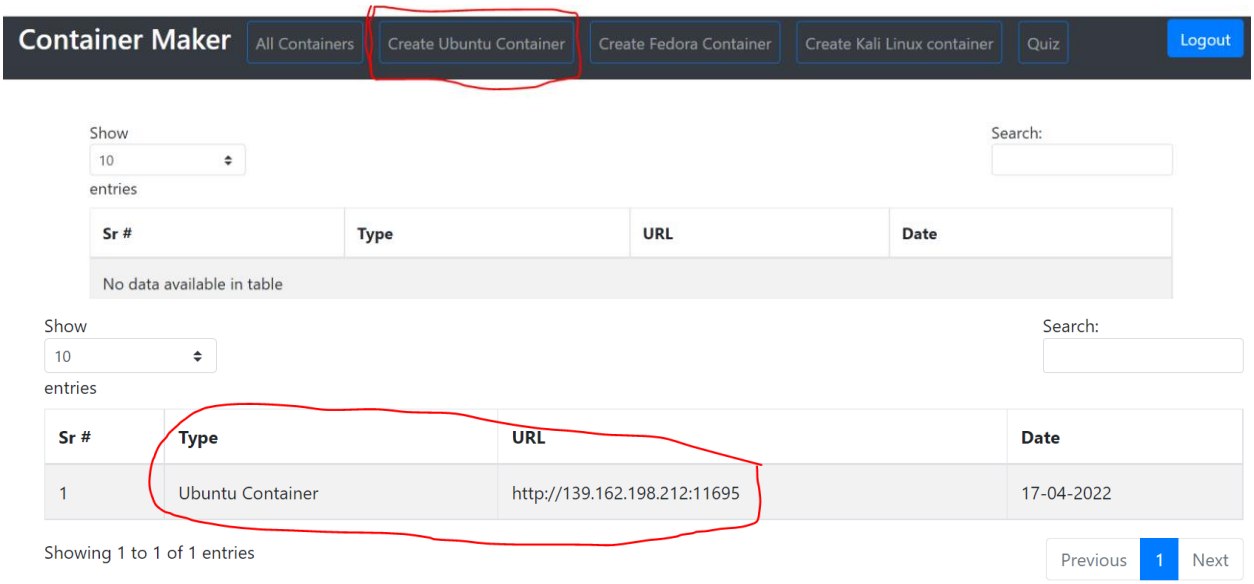


3. Enters login credential. Student email and password if the email type does not exist and registration is complete

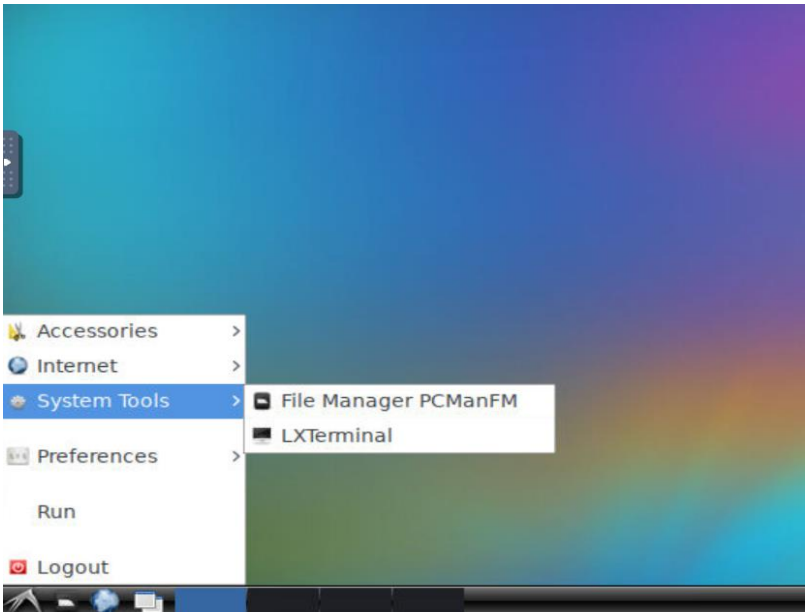
4. Login details are then validated by database
5. If validation is successful, then student will be granted access.
6. Access applications



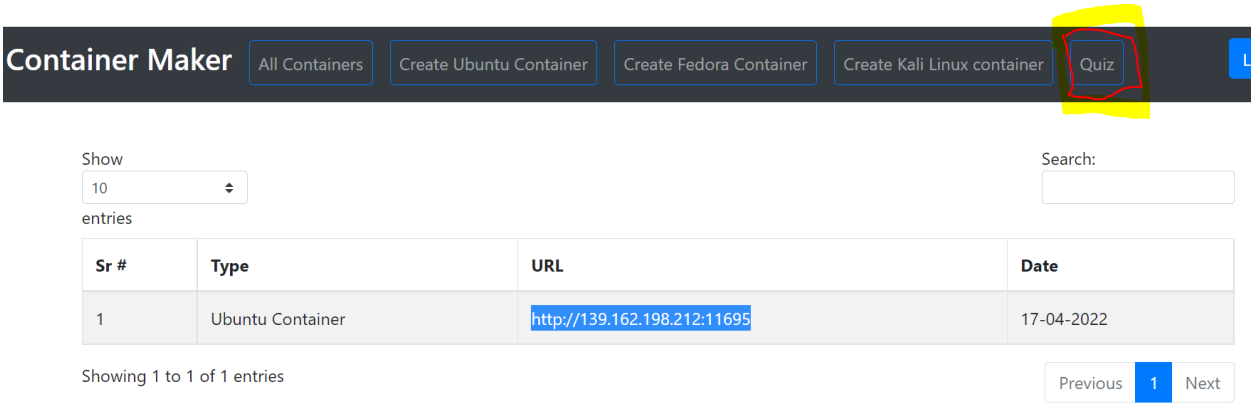
7. Student create VM



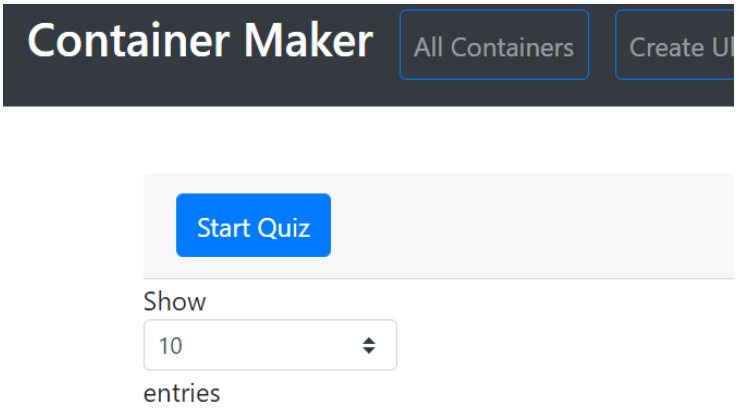
8. Launch VM (Copy link and paste in a new browser)



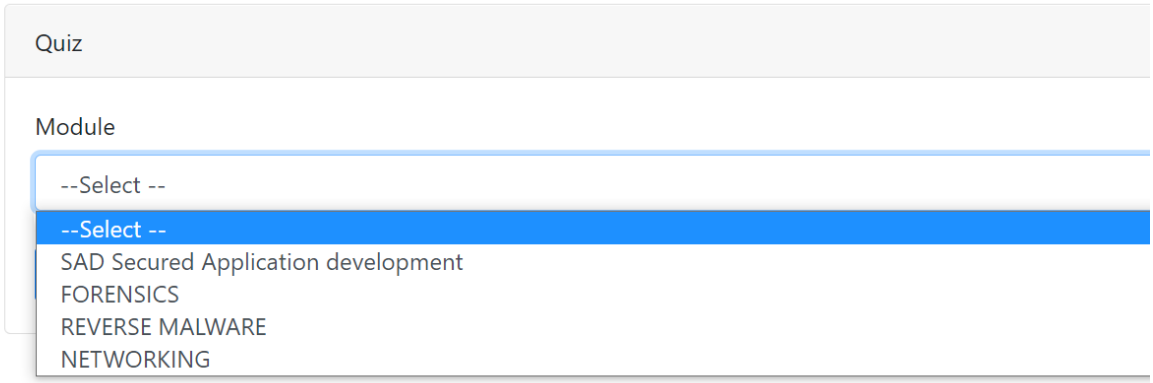
9. Access Quiz



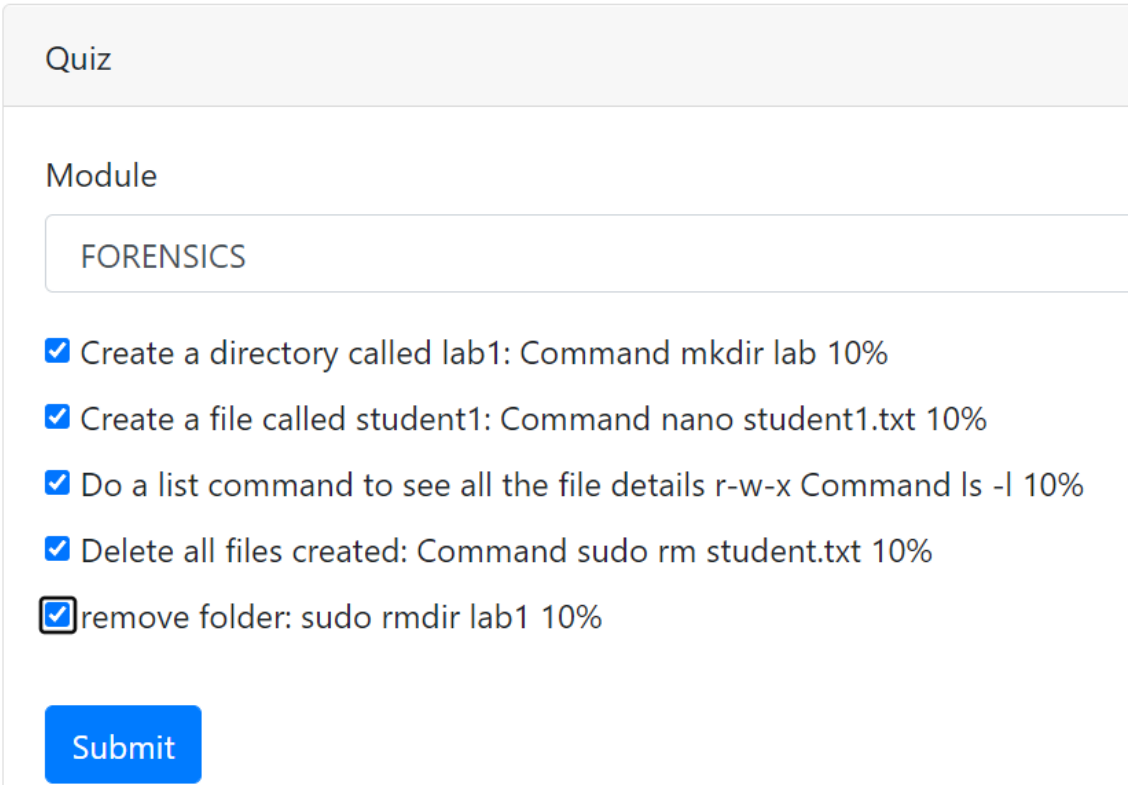
10. Quiz/Module menu



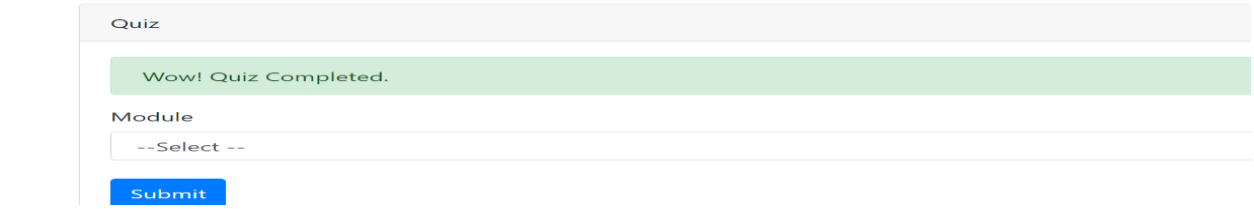
11. Select module/quiz

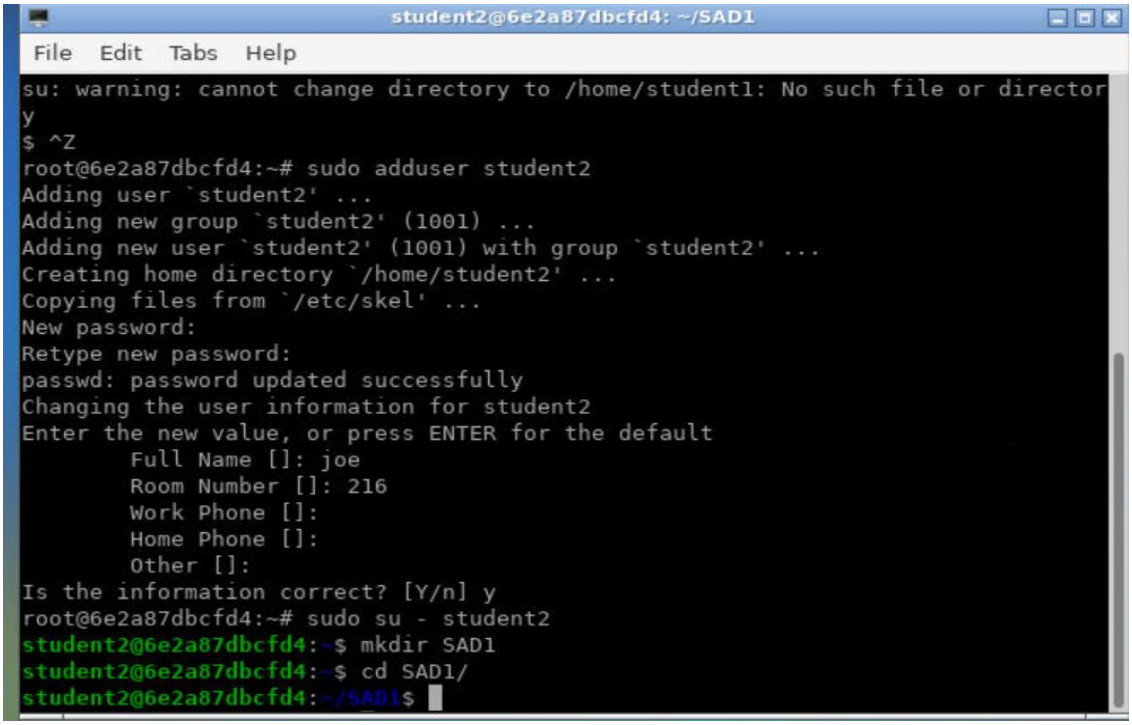


12. Complete tasks and submit

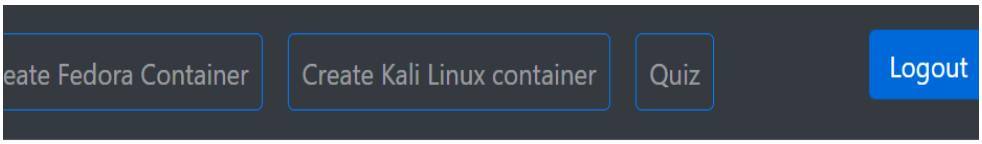


Tutors would confirm completion by checking students VM and if completed tutor will grade. See bullet number 12 in tutors use case.





13. Log out once tutor confirms completion



Result

Successful.

Student has been able to

- Register and login
- Create VM
- Launch VM
- Access Quiz
- Select module/quiz
- Complete tasks and submit
- Log out

Source codes

Config.php

Open — config.php — ~/Documents/test-folder/docker_container/includes

```php
<?php

$server = "localhost";
$user = "root";
$pass = "";
$database = "docker_app";

$conn = mysqli_connect($server, $user, $pass, $database);

if (!$conn) {
   die('Connection Failed');
}
```

```php
include 'includes/config.php';

session_start();

error_reporting(0);

if (isset($_SESSION['id'])) {
   header("Location: dashboard.php");
}
$error='';
if (isset($_POST['submit'])) {
      $email = $_POST['email'];
      $password = md5($_POST['password']);

      $sql = "SELECT * FROM users WHERE email='$email' AND password='$password'";
      $result = mysqli_query($conn, $sql);
      if ($result->num_rows > 0) {
            $row = mysqli_fetch_assoc($result);
            $_SESSION['id'] = $row['id'];
```

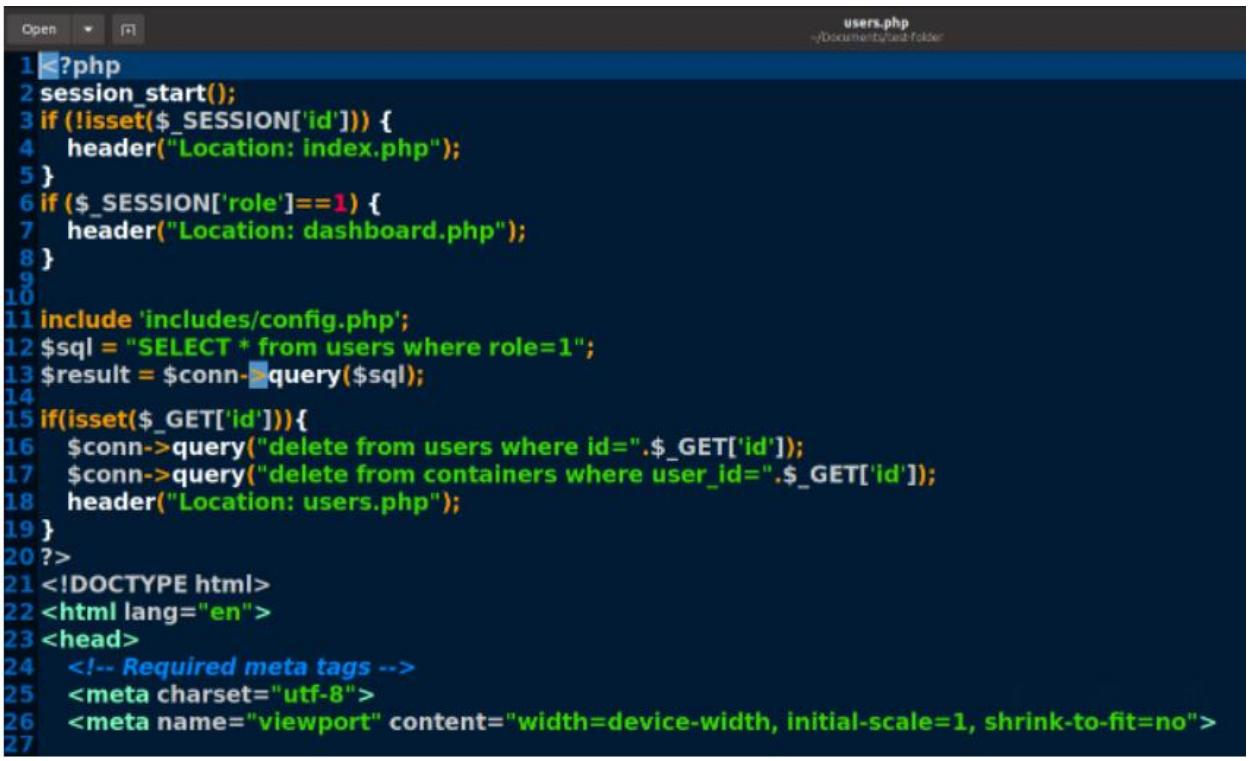
Logout.php

Open — logout.php — ~/Documents/test-folder

```php
<?php

session_start();
session_destroy();

header("Location: index.php");

?>
```

Test.php

Users.php



Register.php

```php
error_reporting(0);

session_start();

if (isset($_SESSION['username'])) {
   header("Location: index.php");
}
$error = '';
if (isset($_POST['submit'])) {
      $username = $_POST['username'];
      $email = $_POST['email'];
      $password = md5($_POST['password']);
      $cpassword = md5($_POST['cpassword']);

      if ($password == $cpassword) {
            $sql = "SELECT * FROM users WHERE email='$email'";
            $result = mysqli_query($conn, $sql);
            if (!$result->num_rows > 0) {
                  $sql = "INSERT INTO users (username, email, password,role)
                              VALUES ('$username', '$email', '$password',1)";
                  $result = mysqli_query($conn, $sql);
                  if ($result) {
            $error.="<p class='alert alert-success'>Wow! User Registration Completed.</p>";
                        $username = "";
                        $email = "";
                        $_POST['password'] = "";
                        $_POST['cpassword'] = "";
                  } else {
            $error.="<p class='alert alert-danger'>Wops! Something Wrong Went.</p>";
                  }
            } else {
           $error.="<p class='alert alert-danger'>Wops! Email Already Exists.</p>";
```

```php
function get_port($id)
{
   $port = "";
   if (strlen($id) == 1 || strlen($id) == 2) {
      $port = (rand(1, 10000));
    }
   if (strlen($id) == 3 | strlen($id) == 4) {
      $port = $id . (rand(1, 100));
    }
   if (strlen($id) == 5 | strlen($id) == 6) {
      $port = (rand(1, 100));
    }
   return $port;
}
```