# Unit Test Report

**Unit**: Doctor Login.

**Source File**: hospital/hms/ doctor/index.php

**Date**: October 16, 2023

**Engineers**: Sai Mukthive, Srikanth Guttikonda, Goutham Bommu, Sai Gunturu

**Automated Test Code:**

***<?php***

***use PHPUnit\Framework\TestCase;***

***function doctor\_login($con, $username, $password) {***

***$hashed\_password = md5($password);***

***$ret = $con->query("SELECT \* FROM doctors WHERE docEmail='$username' and password='$hashed\_password'");***

***$num = $ret->fetch\_assoc();***

***$uip = $\_SERVER['REMOTE\_ADDR'];***

***if ($num) {***

***$\_SESSION['dlogin'] = $username;***

***$\_SESSION['id'] = $num['id'];***

***$uid = $num['id'];***

***$status = 1;***

***$con->query("INSERT INTO doctorslog(uid, username, userip, status) VALUES('$uid', '$username', '$uip', '$status')");***

***return [***

***"message" => "Login successful.",***

***"redirect\_url" => "dashboard.php"***

***];***

***} else {***

***$status = 0;***

***$con->query("INSERT INTO doctorslog(username, userip, status) VALUES('$username', '$uip', '$status')");***

***return [***

***"message" => "Invalid username or password",***

***"redirect\_url" => "index.php"***

***];***

***}***

***}***

***class DoctorLoginTest extends TestCase***

***{***

***private $db\_conn\_mock;***

***protected function setUp(): void***

***{***

***$this->db\_conn\_mock = $this->getMockBuilder(mysqli::class)***

***->disableOriginalConstructor()***

***->getMock();***

***}***

***public function testSuccessfulDoctorLogin()***

***{***

***$queryResultMock = $this->createMock(mysqli\_result::class);***

***$queryResultMock->method('fetch\_assoc')***

***->willReturn([***

***'id' => 123,***

***'specilization' => 'Cardiology',***

***'doctorName' => 'Dr. John Doe',***

***'address' => '123 Medical St, Hospital City',***

***'docFees' => '$150',***

***'contactno' => '1234567890',***

***'docEmail' => 'doctor@example.com',***

***'password' => md5('password123'),***

***'creationDate' => '2023-01-01 10:00:00',***

***'updationDate' => '2023-01-01 10:00:00'***

***]);***

***$this->db\_conn\_mock->expects($this->exactly(2))***

***->method('query')***

***->willReturnOnConsecutiveCalls($queryResultMock, true);***

***$response = doctor\_login($this->db\_conn\_mock, "doctor@example.com", "password123");***

***$this->assertEquals("Login successful.", $response['message']);***

***$this->assertEquals("dashboard.php", $response['redirect\_url']);***

***echo("Test 1: Login successful \n");***

***}***

***public function testUnsuccessfulDoctorLogin()***

***{***

***// Mock the result of the query to simulate a failed login***

***$queryResultMock = $this->createMock(mysqli\_result::class);***

***$queryResultMock->method('fetch\_assoc')->willReturn(null); // No matching doctor found in database***

***$this->db\_conn\_mock->expects($this->exactly(2))***

***->method('query')***

***->willReturnOnConsecutiveCalls($queryResultMock, true); // The second call is for the log insertion***

***$response = doctor\_login($this->db\_conn\_mock, "fakeDoctor@example.com", "wrongPassword");***

***$this->assertEquals("Invalid username or password", $response['message']);***

***$this->assertEquals("index.php", $response['redirect\_url']);***

***echo("Test 2: Unsuccessful login \n");***

***}***

***}***

***?>***

**Inputs:**

Database Connection: Mocked MySQLi connection

Doctor Credentials: Email, Password

**Expected Outputs:**

**a) Successful registration:**

Message: "Login successful ."

RedirectURL: "dashboard.php"

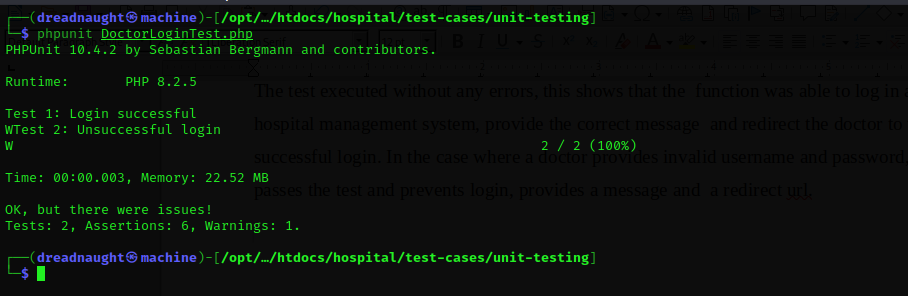
**b) Unsuccessful registration:**

Message: "Invalid username or password ."

RedirectURL: “index.php”

**Actual Outputs:**

The test executed without any errors, indicating that the function was able to log in a doctor to the hospital management system, provide the correct message and redirect the doctor to the right url after a successful login. In the case where a doctor provides invalid username and password, the function passes the test and prevents login, provides a message and a redirect url.



**Test Methodology:**

**Method:**

Mocking and Unit Testing.

**Description:**

The test verifies if the function can successfully log in a doctor with correct credentials: email and password and returns a status message and a redirect URL. It also checks for unsuccessful login attempts and provides the corresponding redirectURL and a status message.

Mocking offers a controlled environment for these tests, ensuring the results are based solely on the function's logic, unaffected by extraneous factors such as server issues or failed database connections.