# Unit Test Report

**Unit**: Doctor Registration.

**Source File**: hospital/hms/admin/add-doctor.php

**Date**: October 16, 2023

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

**Automated Test Code:**

***<?php***

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

***function add\_doctor($con, $specialization, $name, $address, $fees, $contact\_no, $email, $password) {***

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

***$sql = $con->query("INSERT INTO doctors(specilization, doctorName, address, docFees, contactno, docEmail, password) VALUES('$specialization', '$name', '$address', '$fees', '$contact\_no', '$email', '$hashed\_password')");***

***if ($sql) {***

***return [***

***"message" => "Doctor info added Successfully.",***

***"redirect\_url" => "manage-doctors.php"***

***];***

***} else {***

***return [***

***"message" => "Error in adding doctor.",***

***"redirect\_url" => null***

***];***

***}***

***}***

***class AddDoctorTest extends TestCase***

***{***

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

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

***{***

***// Mock the mysqli object***

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

***->disableOriginalConstructor()***

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

***}***

***public function testAddDoctor()***

***{***

***// Mock the mysqli\_query method to simulate successful doctor addition***

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

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

***->willReturn(true);***

***$response = add\_doctor($this->db\_conn\_mock, "Cardiology", "Dr. John Doe", "123 Medical St", "$100", "1234567890", "dr.johndoe@example.com", "password123");***

***$this->assertEquals("Doctor info added Successfully.", $response['message']);***

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

***echo("Test 1: Successful added a doctor to the system \n");***

***}***

***public function testUnsuccessfulAddDoctor()***

***{***

***// Mock the mysqli\_query method to simulate successful doctor addition***

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

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

***->willReturn(false);***

***$response = add\_doctor($this->db\_conn\_mock, "Cardiology", "Dr. John Doe", "123 Medical St", "$100", "1234567890", "dr.johndoe@example.com", "password123");***

***$this->assertEquals("Error in adding doctor.", $response['message']);***

***$this->assertNull( $response['redirect\_url']);***

***echo("Test 2: unable to add a doctor to the system \n");***

***}***

***}***

***?>***

**Inputs:**

Database Connection: Mocked MySQLi connection

Doctor Details: Name, Specialization, Email, Address, Fees, Contact No, Password

**Expected Outputs:**

**a) Successful registration:**

Message: "Registration successful."

RedirectURL: "dashboard.php"

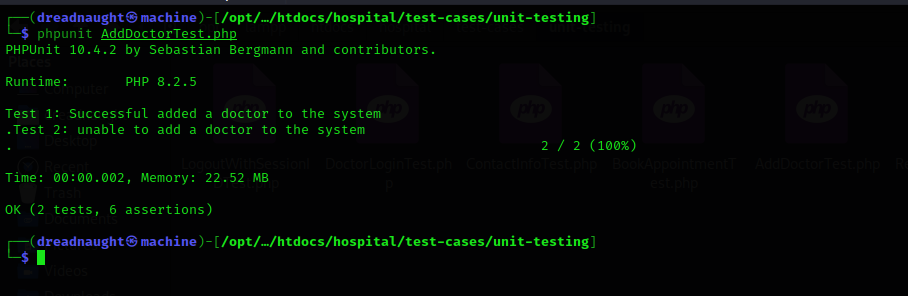
**b) Unsuccessful registration:**

Message: "Registration failed. Email already exists."

RedirectURL: null

**Actual Outputs:**

The test executed without any errors, confirming the function accurately added the doctor's details and provided the expected message and redirect URL. Both successful and unsuccessful test scenarios were considered, ensuring the function not only added data correctly but also handled anomalies and exceptions, returning the anticipated values in each situation.



**Test Methodology:**

**Method:**

Mocking and Unit Testing.

**Description:**

The add\_doctor function is designed to register a new doctor in the system and provide feedback based on the outcome. Using a mock for the MySQLi connection, the function's behavior with the database is simulated without actual database interactions. Mocking offers a controlled environment for these tests, ensuring the results are based solely on the function's logic, unaffected by external factors.

The tests focus on two main scenarios:

1. The function's ability to successfully register a doctor and return the appropriate confirmation message and redirect URL.

2. The function's response when there's a challenge in the registration, such as an existing email in the system or incomplete details.