# Unit Test Report

**Unit**: Register New Patient

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

**Date**: October 30, 2023

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

**Automated Test Code:**

***<?php***

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

***function add\_patient($con, $session\_id, $patname, $patcontact, $patemail, $gender, $pataddress, $patage, $medhis) {***

***// Using the object-oriented style***

***$sql = $con->query("INSERT INTO tblpatient(Docid, PatientName, PatientContno, PatientEmail, PatientGender, PatientAdd, PatientAge, PatientMedhis) VALUES('$session\_id', '$patname', '$patcontact', '$patemail', '$gender', '$pataddress', '$patage', '$medhis')");***

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

***return [***

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

***"redirect\_url" => "add-patient.php"***

***];***

***} else {***

***return [***

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

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

***];***

***}***

***}***

***class RegisterPatientTest extends TestCase***

***{***

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

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

***{***

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

***->disableOriginalConstructor()***

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

***}***

***public function testSuccessfulAddPatient()***

***{***

***// Define patient details***

***$session\_id = "sample\_session\_id123";***

***$patname = "John Doe";***

***$patcontact = "1234567890";***

***$patemail = "john.doe@example.com";***

***$gender = "male";***

***$pataddress = "123 Main St, City, Country";***

***$patage = 30;***

***$medhis = "No known allergies.";***

***// Mock the $mysqli->query() method to simulate successful patient addition***

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

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

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

***// Call the add\_patient function***

***$response = add\_patient($this->db\_conn\_mock, $session\_id, $patname, $patcontact, $patemail, $gender, $pataddress, $patage, $medhis);***

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

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

***echo("Test 1 :Patient added successfully \n");***

***}***

***public function testUnsuccessfulAddPatient()***

***{***

***// Define patient details***

***$session\_id = "sample\_session\_id123";***

***$patname = "Jane Doe";***

***$patcontact = "9876543210";***

***$patemail = "jane.doe@example.com";***

***$gender = "female";***

***$pataddress = "456 Secondary St, City, Country";***

***$patage = 28;***

***$medhis = "Known allergies to penicillin.";***

***// Mock the $mysqli->query() method to simulate unsuccessful patient addition***

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

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

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

***// Call the add\_patient function***

***$response = add\_patient($this->db\_conn\_mock, $session\_id, $patname, $patcontact, $patemail, $gender, $pataddress, $patage, $medhis);***

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

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

***echo("Test 2 :Error adding patient \n");***

***}***

***}***

***?>***

**Inputs:**

The function add\_patient takes in the following inputs:

Database Connection: Mocked MySQli connection

Doctor Details: SESSION ID

Patient Details: Name,Contact,Email,Gender Password, Address,Age,Medical History

**Expected Outputs:**

The function returns an array with two items: a message to the user and a redirect url that will be used by the system to redirect the user.

**a) Successful registration:**

Message: "Registration successful."

RedirectURL: "dashboard.php"

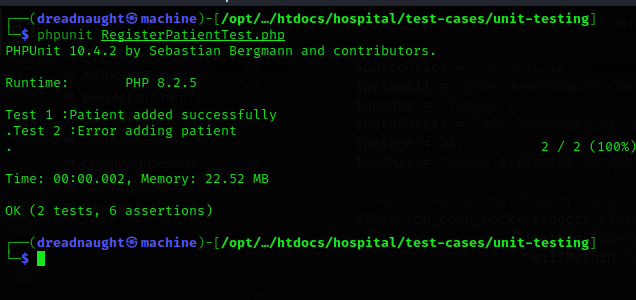
**b) Unsuccessful registration:**

Message: "Registration failed. Email already exists."

RedirectURL: null

**Actual Outputs:**

After executing the test, the function returned the expected message and redirect URL depending on the provided parameters and their completeness and passed the test without errors.



**Test Methodology:**

**Method:**

Mocking and Unit Testing.

**Description:**

The register\_patient function adds a new patient into the database and return a message that indicates the success or failure of this action. During the testing, a simulated MySQLi connection was employed.

The test verifies the capability of the function to correctly register a patient and subsequently deliver the expected acknowledgment message and redirection path. An additional test ensures that the function can effectively handle situations where an existing email is entered, and it then returns the appropriate error message.

This mocking technique in the testing ensures a controlled setting, isolating the function from external dependencies. This ensures that the outcomes of the tests are strictly based on the function's internal logic and are not affected by outside factors.