# Unit Test Report

**Unit**: Contact Us Module.

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

**Date**: October 28, 2023

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

**Automated Test Code:**

***<?php***

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

***function submit\_contact\_info($name, $email, $mobileno, $dscrption, $con) {***

***$stmt = $con->prepare("INSERT INTO tblcontactus(fullname, email, contactno, message) VALUES (?, ?, ?, ?)");***

***$stmt->bind\_param("ssss", $name, $email, $mobileno, $dscrption);***

***// Execute the query***

***if ($stmt->execute()) {***

***// If the insertion was successful***

***return [***

***"message" => "Your information was successfully submitted.",***

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

***];***

***} else {***

***// If there was an error in insertion***

***return [***

***"message" => "There was an error in submitting your information.",***

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

***];***

***}***

***}***

***class ContactInfoTest extends TestCase***

***{***

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

***private $stmt\_mock;***

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

***{***

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

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

***->disableOriginalConstructor()***

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

***// Mock the prepared statement***

***$this->stmt\_mock = $this->getMockBuilder(mysqli\_stmt::class)***

***->disableOriginalConstructor()***

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

***// Set the return value for the prepare method***

***$this->db\_conn\_mock->method('prepare')***

***->willReturn($this->stmt\_mock);***

***}***

***// stimulate successful data insertions***

***public function testSuccessfulContactInfoSubmission()***

***{***

***$name = "John Doe";***

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

***$mobileno = "1234567890";***

***$dscrption = "Test message";***

***// Mock the bind and execute methods***

***$this->stmt\_mock->expects($this->once())***

***->method('bind\_param')***

***->with("ssss", $name, $email, $mobileno, $dscrption)***

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

***$this->stmt\_mock->expects($this->once())***

***->method('execute')***

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

***$response = submit\_contact\_info($name, $email, $mobileno, $dscrption, $this->db\_conn\_mock);***

***$this->assertEquals("Your information was successfully submitted.", $response['message']);***

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

***echo("Test 1: Successfully added contact message \n");***

***}***

***// stimulating unssuccessful data insertion***

***public function testUnsuccessfulContactInfoSubmission()***

***{***

***$name = "Jane Doe";***

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

***$mobileno = "0987654321";***

***$dscrption = "Failed test message";***

***// Mock the bind and execute methods***

***$this->stmt\_mock->expects($this->once())***

***->method('bind\_param')***

***->with("ssss", $name, $email, $mobileno, $dscrption)***

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

***$this->stmt\_mock->expects($this->once())***

***->method('execute')***

***->willReturn(false); // This will simulate a failed insertion***

***$response = submit\_contact\_info($name, $email, $mobileno, $dscrption, $this->db\_conn\_mock);***

***$this->assertEquals("There was an error in submitting your information.", $response['message']);***

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

***echo("Test 2: Unable to add contact message \n");***

***}***

***}***

***?>***

**Inputs:**

Database Connection: Mocked MySQLi connection.

Patient Details: Name,Email,Mobile Number, Description, Database Connection

**Expected Outputs:**

The function submit\_contact\_info returns an array containing a message and a redirectURl. The URL and message changes depending on whether the action was successful or not.

**a) Successful registration:**

Message: "Your information was successfully submitted."

RedirectURL: "index.php"

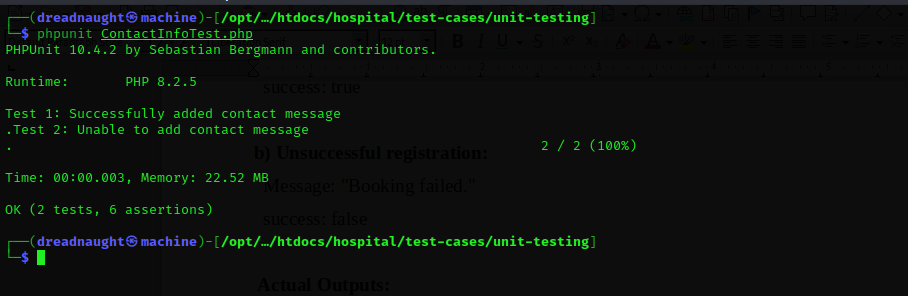
**b) Unsuccessful registration:**

Message: "There was an error in submitting your information."

RedirectURL: null

**Actual Outputs:**

After executing the test, the function returned the expected message and redirect URL without any errors indicating that the contact message has been successfully recorded and the function works as expected. This function worked as expected both in successful and unsuccessful message sending scenarios.



**Test Methodology:**

**Method:**

Mocking and Unit Testing.

**Description:**

The function submit\_contact\_info was designed to insert contact information into a database and return a message based on the success or failure of that operation. For the test, both the MySQLi connection and the prepared statement were mocked to simulate the behavior of the actual database interaction without making real database changes.

The test case checks if the function can successfully insert contact information and return the expected success message and redirect URL. Mocking was chosen as it offers a controlled environment for these tests, ensuring the results are based solely on the function's logic and removes the impact of extraneous factors on the test.