# Unit Test Report

**Unit**: Booking an Appointment .

**Source File**: hospital/hms/book-appointment.php

**Date**: October 13, 2023

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

**Automated Test Code:**

***<?php***

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

***function book\_appointment($con, $specialization, $doctorid, $fees, $appdate, $apptime, $session\_id) {***

***$userstatus = 1;***

***$docstatus = 1;***

***$query = $con->query("INSERT INTO appointment(doctorSpecialization, doctorId, userId, consultancyFees, appointmentDate, appointmentTime, userStatus, doctorStatus) VALUES('$specialization', '$doctorid', '$session\_id', '$fees', '$appdate', '$apptime', '$userstatus', '$docstatus')");***

***if ($query) {***

***return [***

***"message" => "Your appointment successfully booked.",***

***"success" => true***

***];***

***} else {***

***return [***

***"message" => "Booking failed.",***

***"success" => false***

***];***

***}***

***}***

***class BookAppointmentTest 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 testBookAppointment()***

***{***

***// Mock the mysqli\_query method to simulate successful appointment booking***

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

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

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

***$response = book\_appointment($this->db\_conn\_mock, "Cardiology", "doc123", "$100", "2023-11-10", "10:00 AM", "session\_id123");***

***$this->assertEquals("Your appointment successfully booked.", $response['message']);***

***$this->assertTrue($response['success']);***

***echo("Test 1: Successfully booked an appointment \n");***

***}***

***public function testFailedBookAppointment()***

***{***

***// Mock the mysqli->prepare method to simulate failed appointment booking***

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

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

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

***$response = book\_appointment($this->db\_conn\_mock, "Cardiology", "doc123", "$100", "2023-11-10", "10:00 AM", "session\_id123");***

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

***$this->assertFalse($response['success']);***

***echo("Test 2:Unable to book an appointment \n");***

***}***

***}***

***?>***

**Inputs:**

The function takes the following inputs:

Database Connection: Mocked MySQLi connection.

Doctor Details: Specialization,Doctor ID and Consultancy Fees.

Appointment Date: Date of the appointment.

Appointment Time: Time slot for the appointment.

User Session ID: A unique session identifier.

**Expected Outputs:**

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

**a) Successful registration:**

Message: "Your appointment successfully booked."

success: true

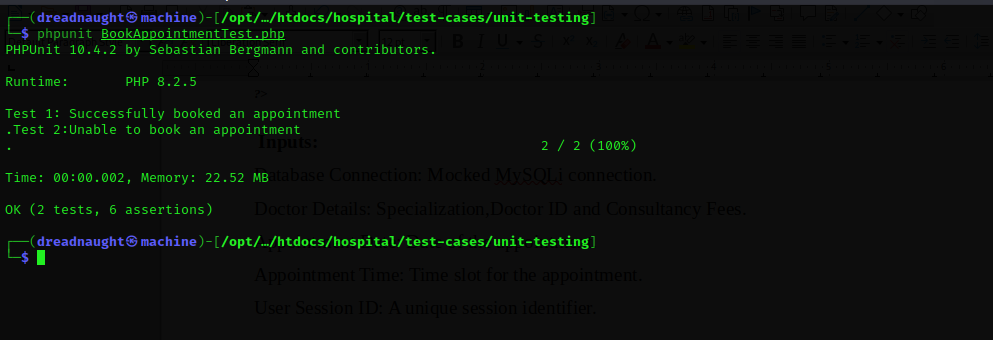
**b) Unsuccessful registration:**

Message: "Booking failed."

success: false

**Actual Outputs:**

The test executed without any errors, confirming the function works as expected and books an appointment when the correct details are provided and returns an error message when incorrect or incomplete data is provided.



**Test Methodology:**

**Method:**

Mocking and Unit Testing.

**Description:**

The book\_appointment function books an appointment by inserting details into the appointment table.

In the test, the MySQLi connection is mocked to simulate the behavior of the database interaction.

The test verifies if the function can successfully book an appointment and return the expected success message.

Mocking offers a controlled environment for these tests, ensuring the results are based solely on the function's logic, unaffected by external factors.