**Sprint Three**

**testAddDoctor**

The test function creates a new doctor using the interface. After it is entering the details, it will then click the submit button and check a function called isSuccessful(). This function returns true or false depending on if a doctor has been successfully added.

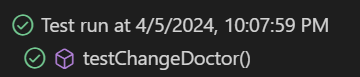
**A black background with white text

Description automatically generatedTest output:**

**testChangeDoctor**

The test function opens up DoctorSelectionPopup and click the button. This will simulate changing a doctor. The test function then check the SQL database to see if the doctor has been updated to Taylor Paddington. If they have, then the test result will be passed if not, then fail.

**Test output:**

****

**testAddPatient**

The test function simulates adding a new patient. It inputs all the necessary information. After the test class clicks the submit button then it will check the database to see if the newly added patient is present. If they are present then it will pass the test, if not then it will fail.

**Test output:**

**A close up of a black background

Description automatically generated**

**testViewDoctorScreen**

The test function checks if the doctorscreen is successfully displaying information inside the table. Since there is sample data already in the database, we just have to run a query that checks the row count of a table. If it is less than 1, it will fail as the function is not correctly displaying the doctors. If it passes then it mean that the doctors are being shown on the screen.

**Test output:**

**A close up of a black background

Description automatically generated**

**testRoleBasedAccessAdmin**

The test function checks if certain buttons are visible. As we are testing for the Admin role, the admin should see every single button. So if it fails the test, this means that one of the buttons are not being shown but if it passes the test then it means that all the buttons are being shown.

**Test output:**

**A black background with white text

Description automatically generated**

**testBookingAndConflictingBooking**

The test function checks if the arrangebooking feature works. So it creates a booking and submits it. The test class will check if the booking was successfully created. The test function will then submit the same booking again to check if that the conflicting booking functionality works. If it does then it will pass, if it doesn’t it will fail.

**Test output:**

**A black background with white text

Description automatically generated**

**testPastBooking**

The test function tries creating a booking in the past. So the function tries to make a booking for 25th of November 2020. The test function check what the text says above the submit button. If it returns an error saying that you cannot select a past date.

**A black background with white text

Description automatically generated**

**testRescheduleBooking**

The test function tries and reschedule a preexisting booking. It will try and move it from 15th April 2024 to the 16th April 2024 9:15am. The test function calls the function that deals with rescheduling. After the function is finished, the test function will run an query to see if it has been changed accordingly. If it returns pass that means it was successfully changed, if it fails then it makes there is something wrong with the rearrangeBooking function.

A black background with white text

Description automatically generated