This project underwent a combination of both manual and unit testing to ensure software completion. The unit testing phase was used to isolate specific methods in the application and evaluate their functionality. The provided unit tests focus on the feature of creating a new appointment and a few associated validation checks. These specific tests cover ensuring appointments can be scheduled with valid data, appointments can only be scheduled in the future, and scheduled appointments can not end before they start.

Beyond unit tests, a significant portion of the testing process was conducted manually. This involved me inputting data confirming the software has the correct responses. This method of testing was particularly crucial for functionalities that required complex scenarios that were challenging to emulate with automated tests. For example testing of several functions within the software would require a testing database to be created and filled with mock data. This would be significantly more complicated than testing those features manually using the current database. Here is a list containing the majority of the features that were tested manually: Overlapping appointment validation, report generation, edit appointment, delete appointment, search functionality, secure login, secure logout, phone number validation, name validation, authentication timer, and print report feature.

**Unit Tests including screenshots, scripts, and results:**

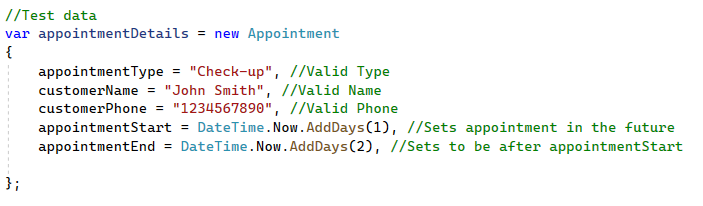
**Test Objective:**

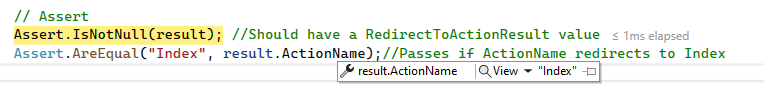
The main purpose of this unit test is to validate that the appointment creation logic works as expected.

**Test Scenarios:**

**Scenario 1: Valid Appointment Creation**

**Input:** Provide valid appointment details.

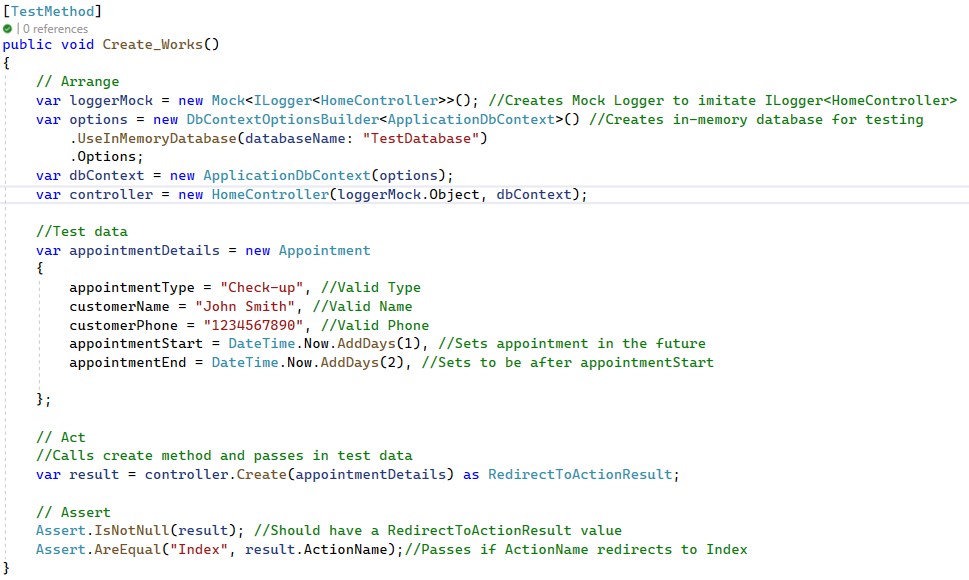
**Expected Output:** The appointment is created successfully. We know the appointment is created successfully if it redirects to Index.



**Scenario 1 Unit Test Success:**

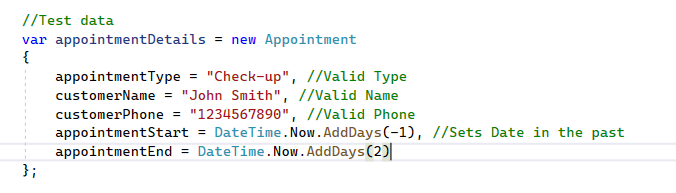


**Scenario 1 Entire Script With Comment Explanations:**

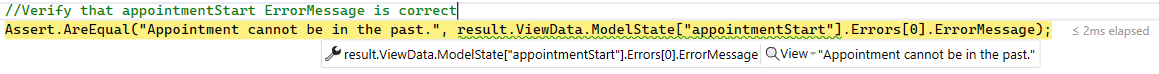
****

**Scenario 2: Check That Appointment is not in the Past**

**Input:** Provide appointment details with a past start date. DateTime.Now sets date to the current day .AddDays(-1) removes one day to set appointmentStart value in the past.



**Expected Output:** An error message should be displayed. In this example it checks to see if the ErrorMessage matches for the ModelState[“appointmentStart”]. Since it matches it proves that the validation check for appointments being in the past is working correctly.



**Scenario 2 Unit Test Success:**

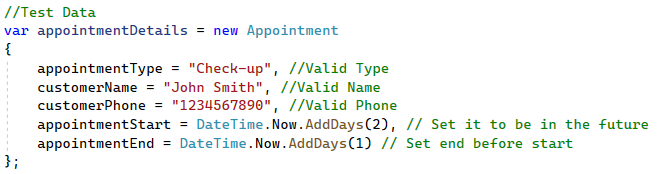
****

**Scenario 2 Entire Script With Comment Explanations:**

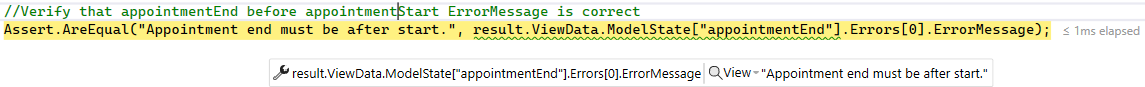
****

**Scenario 3: End Date Before Start Date**

**Input:** Provide appointment details where the end date is before the start date. Similar to the previous example I use DateTime.Now to get the current Date but this time I add 2 days to the appointmentStart and 1 day to appointmentEnd making the appointmentEnd before the appointmentStart.



**Expected Output:** An error message should be displayed. Just like the last example it checks to see if the ErrorMessage matches but this time for the ModelState[“appointmentEnd”]. Since it matches it proves that the validation check does not allow for appointments to end before they start.



**Scenario 3 Unit Test Success:**

****

**Scenario 3 Entire Script with Comment Explanations:**

****