

ST10445830

## Unit Tests

Test Explorer

Test run finished: 1 Tests (1 Passed, 0 Failed, 0 Skipped) run in 229 ms

0 Warnings 0 Errors

Test	Duration	Traits	Error Message
claimtest (6)	1,6 sec		
claimtest (6)	1,6 sec		
ClaimTests (6)	1,6 sec		
AdditionalNotes	394 ms		
CalculatedTotalAmount	530 ms		
FileUploadProperty	666 ms		
StatusPropertyApproved	9 ms		
StatusPropertydefault	7 ms		
StatusPropertyRejected	6 ms		

Run Debug

Test Detail Summary

- claimtest.ClaimTests.
- Source: [ClaimTe](#)
- Duration: 9 ms

```
claimtest claimtest.ClaimTests
1 using System.IO;
2 using Microsoft.AspNetCore.Http;
3 using Moq;
4 using Xunit;
5 using CMCS_ST10445830.Models;
6
7 namespace claimtest
8 {
9     0 references
10     public class ClaimTests
11     {
12         //Check if total amount is calculated correctly
13         [Fact]
14         0 references
15         public void CalculatedTotalAmount()
16         {
17             //arrange phase
18             var claim = new Claim();
19
20             claim.HoursWorked = 10; //will set hours to 10
21             claim.HourlyRate = 45; //set rate to 45
22
23             //act phase
24
25             var getResult = claim.CalculateTotalAmount();
26
27             //assert phase
28             Assert.Equal(450, getResult);
29         }
30
31         //check if Notes property can be set and retrieved correctly
32         [Fact]
33         0 references
34         public void AdditionalNotes()
35         {
36             //arrange phase
37             var claim = new Claim();
38             claim.Notes = "This is a test note for the claim.";
39
40             //act phase
41             var getResult = claim.Notes;
42
43             //assert phase
44             Assert.Equal("This is a test note for the claim.", getResult);
45         }
46     }
47 }
```

NuGet: claimtest   Claim.cs   ClaimTests.cs\*   Program.cs

claimtest   claimtest.ClaimTests

```
43
44
45 //Check if file upload property works correctly
46 [Fact]
47 0 references
48 public void FileUploadProperty()
49 {
50     //arrange phase
51     var claim = new Claim();
52
53     // Create a mock IFormFile object
54     var fileMock = new Mock<IFormFile>();
55     fileMock.Setup(f => f.FileName).Returns("invoice.pdf");
56
57     //act phase
58     claim.DocumentFile = fileMock.Object;
59     claim.DocumentUrl = "/uploads/invoice.pdf";
60
61     //assert phase
62     Assert.Equal("invoice.pdf", claim.DocumentFile.FileName);
63 }
64
65 //Check if status is Approved
66 [Fact]
67 0 references
68 public void StatusPropertyApproved()
69 {
70     //arrange phase
71     var claim = new Claim();
72     //act phase
73     claim.Status = "Approved";
74     //assert phase
75     Assert.Equal("Approved", claim.Status);
76 }
77
78 //Check if default status is "Pending"
79 [Fact]
80 0 references
81 public void StatusPropertydefault()
82 {
83     //arrange phase
84     var claim = new Claim();
85     //act phase
86     var getResult = claim.Status;
87     //assert phase
88     Assert.Equal("Pending", getResult);
89 }
```

```
6
7
8 //Check if status is Rejected
9 [Fact]
10 0 references
11 public void StatusPropertyRejected()
12 {
13     //arrange phase
14     var claim = new Claim();
15     //act phase
16     claim.Status = "Rejected";
17     //assert phase
18     Assert.Equal("Rejected", claim.Status);
19 }
20
21 }
```