1.12. DSD 605 Software Testing and Security- Assessment 1

Module Code	DSD 605	**Module Title**	Software Testing and Security		
Level	6	**Credit Value of Module**	15		
Assessment Task	Using an existing project design and execute Security Tests and Unit Tests on pure methods. Create a test report.				
Assessment No	1	**Total marks**	40	**Weighting**	40%
Type of Submission					
Method of Submission					
Due Date	XXXXX				

Learning Outcomes addressed

Learning outcomes:

LO1:

Design a range of tests including, Unit testing and Security testing to provide technical support

LO2:

Select and apply Software testing and quality assurance methods including technical documentation

Passing Criteria

DSD 605 Software Testing and Security

Total Marks: 40
Total Weighting: 40%

A+	90-100%	C+	60-64%
A	85-89%	C	55-59%
A-	80-84%	**C-**	**50-54% PASS MARK**
B+	75-79%	D+	45-49%
B	70-74%	D	40-44%
B-	65-69%	D-	0-39%

Resubmission or Reassessment Requirements

If a student performance fails to meet any of the requirements in the marking schedule, the student may be given opportunity to resubmit or be reassessed.

Resubmission: If it is apparent (before or during marking) that evidence is incomplete, either by omission or error, the assessor may direct the student's attention to the areas of breach, in general terms, and invite him/her to resubmit the work with the identified areas corrected. Only one resubmission opportunity may be given.

Reassessment: If, after valid processing has been applied, a student is still judged not to have reached the standard required, the student may be offered a further assessment opportunity. A new date will be agreed upon, at the assessor's discretion. Only one reassessment opportunity may be given.

1.12. DSD 605 Software Testing and Security- Assessment 1

Assessment brief and instructions

Instruction for the assessment

- This assessment is worth 40% of the total marks for the course.
- The assessment must be delivered individually.
- Your submission **must** include references (APA 7 format) if referring to the source material. Failure to do so will have serious consequences.
- You must fill out and submit the ATC Vision College cover sheet.

Assessment Conditions

This is a resource-based assessment. This means that you may have access to any relevant resources to assist you.

This could include your learning materials, information on the Internet, and so on. However, all work must be your own with no assistance from any other person.

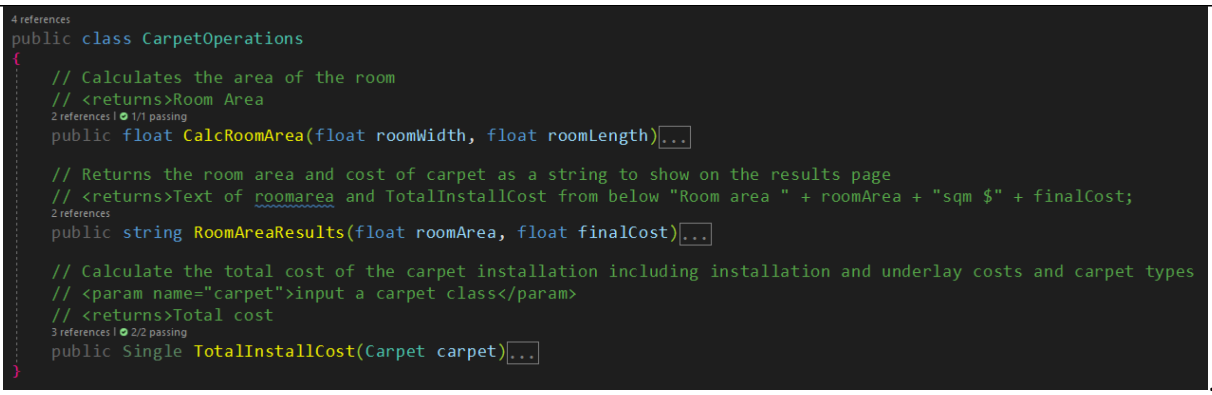
Assessment brief

Using an existing ASP.net Core project, design Unit Tests on pure methods. Generate the test report.

Remember to create a new GitHub repository so you don't overwrite your old one.

Unit Testing

1. Open the Carpet Calculator project.
2. Modify your Carpet Operations class to process the three methods below.

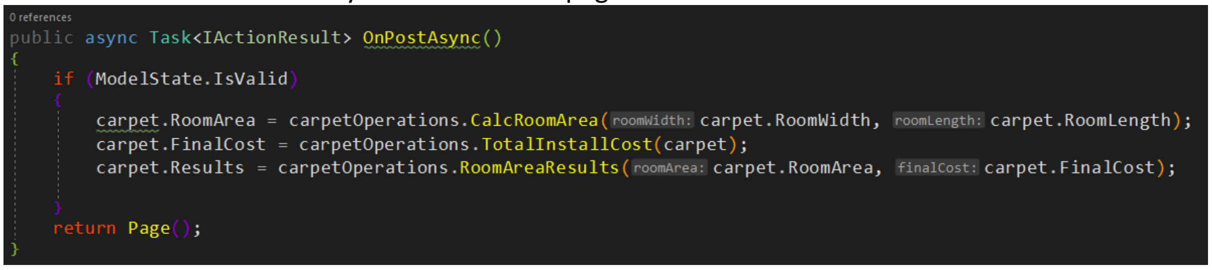


```
4 references
public class CarpetOperations
{
    // Calculates the area of the room
    // <returns>Room Area
    2 references | 1/1 passing
    public float CalcRoomArea(float roomWidth, float roomLength)...

    // Returns the room area and cost of carpet as a string to show on the results page
    // <returns>Text of roomarea and TotalInstallCost from below "Room area " + roomArea + "sqm $" + finalCost;
    2 references
    public string RoomAreaResults(float roomArea, float finalCost)...

    // Calculate the total cost of the carpet installation including installation and underlay costs and carpet types
    // <param name="carpet">input a carpet class</param>
    // <returns>Total cost
    3 references | 2/2 passing
    public Single TotalInstallCost(Carpet carpet)...
}
```

Consume those methods on your Index.cshtml page as below

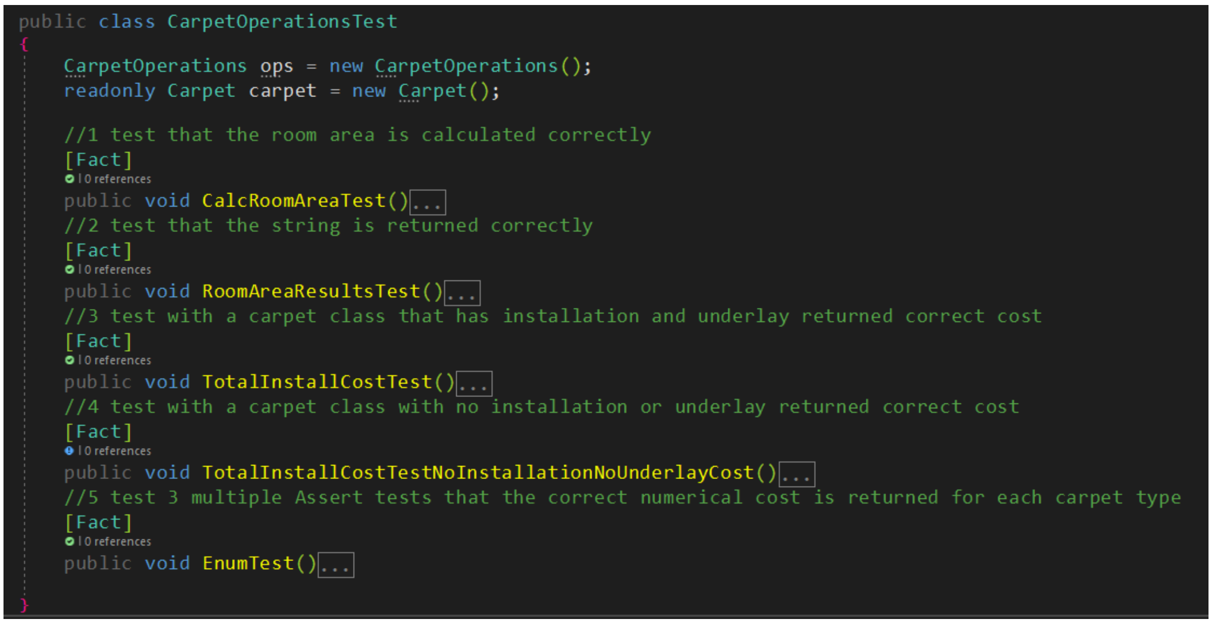


```
0 references
public async Task<IActionResult> OnPostAsync()
{
    if (ModelState.IsValid)
    {
        carpet.RoomArea = carpetOperations.CalcRoomArea(roomWidth: carpet.RoomWidth, roomLength: carpet.RoomLength);
        carpet.FinalCost = carpetOperations.TotalInstallCost(carpet);
        carpet.Results = carpetOperations.RoomAreaResults(roomArea: carpet.RoomArea, finalCost: carpet.FinalCost);

    }
    return Page();
}
```

3. Create the following Unit tests for each of the pure methods.
    1. Test that the Room area is calculated correctly. Pass in RoomWidth and RoomLength and return RoomArea
    2. Test that the string is returned correctly. Pass in RoomArea, FinalCost and return Results.
    3. Test with the carpet class with added values, that **with** the HasInstallation = true and HasUnderlay = true are returning the correct values.
    4. Test with the carpet class with added values, that **without** the HasInstallation = false and HasUnderlay = false the method is returning the correct values.

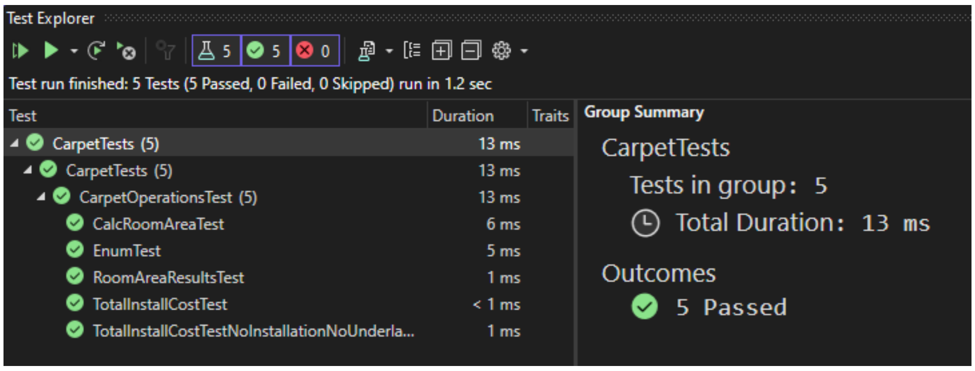
5. Test that the carpet enums are returning the correct numerical values. Use Assert Multiple to include all three tests into one.



```
public class CarpetOperationsTest
{
    CarpetOperations ops = new CarpetOperations();
    readonly Carpet carpet = new Carpet();

    //1 test that the room area is calculated correctly
    [Fact]
    | 0 references
    public void CalcRoomAreaTest()...
    //2 test that the string is returned correctly
    [Fact]
    | 0 references
    public void RoomAreaResultsTest()...
    //3 test with a carpet class that has installation and underlay returned correct cost
    [Fact]
    | 0 references
    public void TotalInstallCostTest()...
    //4 test with a carpet class with no installation or underlay returned correct cost
    [Fact]
    | 0 references
    public void TotalInstallCostTestNoInstallationNoUnderlayCost()...
    //5 test 3 multiple Assert tests that the correct numerical cost is returned for each carpet type
    [Fact]
    | 0 references
    public void EnumTest()...
}
```

,



Test Explorer

Test run finished: 5 Tests (5 Passed, 0 Failed, 0 Skipped) run in 1.2 sec

Test	Duration	Traits
CarpetTests (5)	13 ms	
CarpetTests (5)	13 ms	
CarpetOperationsTest (5)	13 ms	
CalcRoomAreaTest	6 ms	
EnumTest	5 ms	
RoomAreaResultsTest	1 ms	
TotalInstallCostTest	< 1 ms	
TotalInstallCostTestNoInstallationNoUnderla...	1 ms	

Group Summary
CarpetTests
Tests in group: 5
Total Duration: 13 ms
Outcomes
5 Passed

In your Answer Sheet screenshot the following:

1. The 5 methods, (including the Enum method)
2. The 5 tests
3. The Test Explorer.

4. Using an existing Kiwi Quiz ReactJS project, design the following tests. This may require you to modify your code to produce pure functions for the first two functional tests. The following 6 are UI tests.

Using Jest for checking functions.

1. Check the CheckForWinnerLoser function returns **Loser** for different inputs. If you pass in two unequal words, does it return false. This might require some rebuilding of your function so that it becomes a pure function.

2. Check the CheckForWinnerLoser function returns **Winner** for different inputs. If you pass in two unequal words, does it return false. This might require some rebuilding of your function so that it becomes a pure function.

Using Testing Library for UI testing

Suite Title = "**Test the Footer Component**"

Create a gameData object with values Q: "Question", and A: "Answer",

3. Test Title: Is The Google link displaying? Test that "Google Answer" is on the page.
4. Test Title: s the Hint showing? Test that "Hint: Question" is on the page.

Suite Title = " **Test the Header Component**"

Create a gameData object with values Q: "Start", and A: "Auckland",

5. Test Title: Is The Kiwi quiz displaying? Test that "The Kiwi quiz" is on the page.
6. Test Title: Is the question displaying from the gameData? Test that "Start" is on the page.

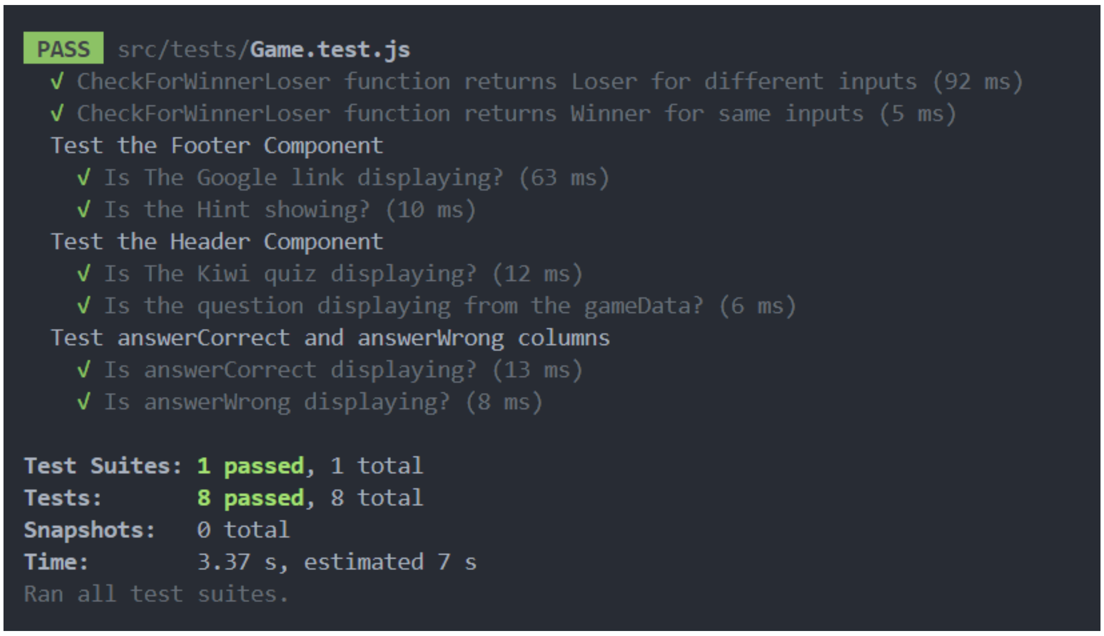
Suite Title = " Test answerCorrect and answerWrong columns"

Using the ResultsPage that displays the results. Create two arrays of data

```
fakeAnswerCorrect = ["correct", "correct2", "correct3"];
fakeAnswerWrong = ["incorrect", "incorrect2", "incorrect3"];
```

Pass them into the Results function and check that they all displaying on the page.

7. Test Title: Is answerCorrect displaying? Test that the array you pass in is displaying on the page.
8. Test Title: Is answerWrong displaying? Test that the array you pass in is displaying on the page.



```
PASS  src/tests/Game.test.js
  √ CheckForWinnerLoser function returns Loser for different inputs (92 ms)
  √ CheckForWinnerLoser function returns Winner for same inputs (5 ms)
  Test the Footer Component
    √ Is The Google link displaying? (63 ms)
    √ Is the Hint showing? (10 ms)
  Test the Header Component
    √ Is The Kiwi quiz displaying? (12 ms)
    √ Is the question displaying from the gameData? (6 ms)
  Test answerCorrect and answerWrong columns
    √ Is answerCorrect displaying? (13 ms)
    √ Is answerWrong displaying? (8 ms)

Test Suites: 1 passed, 1 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        3.37 s, estimated 7 s
Ran all test suites.
```

1.12. DSD 605 Software Testing and Security- Assessment 1

In your Answer Sheet screenshot the following:

1. The 8 tests
2. The Test result.

4. Execute a Security Test of your choice, that is the best suitable for your application, for both your React project and your Asp.net project.

Choose from the following tests:

- Cross-Site Scripting (XSS) tests:
- Cross-Site Request Forgery (CSRF) tests:
- Input Validation tests:
- Penetration Testing:

1. Screenshot your two tests.
2. Explain the type of tests you have used
3. Explain the output you generated and what would you would expect if there was a security risk.

As well

Create a small video showing both the projects running their tests.

MARKING RUBRIC

Assessment Acceptance criteria	**Grade Distribution**			**Student Mark**	**Comments**
	Incorrect or not attempted	**Partially correct**	**Fully correct**		
Asp.net Unit Tests					
1. Test that the Room area is calculated correctly.	0	2	2		
2. Test that the string is returned correctly.	0	2	**2**		
3. Test with the carpet class with added values, that with the HasInstallation = true and HasUnderlay = true are returning the correct values.	0	1	**2**		
4. Test with the carpet class with added values, that without the HasInstallation = false and HasUnderlay = false the method is returning the correct values.	0	1	2		
5. Test that the carpet enums are returning the correct numerical values.	0	2	3		
React JS Unit Tests					
1. Check the CheckForWinnerLoser function returns Loser for different inputs	0	2	3		
2. Check the CheckForWinnerLoser function returns Winner for different inputs	0	2	4		
3. Test Title: Is The Google link displaying? Test that “Google Answer” is on the page.	0	1	2		
4. Test Title: s the Hint showing? Test that “Hint: Question” is on the page.	0	1	2		
5. Test Title: Is The Kiwi quiz displaying? Test that “The Kiwi quiz” is on the page.	0	1	2		
6. Test Title: Is the question displaying from the gameData? Test that “Start” is on the page.	0	1	2		
7. Test Title: Is answerCorrect displaying? Test that the array you pass in is displaying on the page.	0	2	5		
8. Test Title: Is answerWrong displaying? Test that the array you pass in is displaying on the page.	0	2	5		
Execute two Security Test					
1. Security Test 1	0		2		
2. Security Test 2			2		
		Total 40 Marks		**/40**	

NOTES:

***Sufficient** is defined as per the points below:

a) The answer provided by the student is correct and is supported by relevant ideas and/or supporting details.
b) Information is presented in a logical manner that is easily followed.

MARKING RUBRIC

c) There is minimal interruption to the work due to misspellings and/or grammatical errors.

*__Insufficient__ is defined as not meeting any of the points above (a, b and c).

*__Reasonable__ is defined as meeting at least point a) above.