**SQPT In Class Mock**

**Instructions:**

1. Along with this document file there is a zip project file, which needs to be downloaded and unzipped.
2. Based on the project source code, you must create possible functional requirements, traceability matrix and test cases in a word document file.
3. Inside of a project there is a TestProject2 under which you can find a file UnitTest1.cs, you are required to write your unit test on this file. To submit, you need to zip it back to a single file (0000XXStudentID.zip) and upload it back to LMS into in class assessment of SQPT Module.
4. Please, include your resulting word document into above 0000XXStudentID.zip file as well.

***Task #1:*** *Create* ***ten functional requirements*** *for a console-based Recipe Management System where users can create, edit, and delete recipes. Include features like search, detailed info. (see attached project source code for reference)*

***Total: 10 points.***

**ANSWER:**

Functional Requirements

|  |  |
| --- | --- |
| FR ID | Description |
| FR01 | Users should be able to create new recipe |
| FR02 | User should be able to edit a recipe by ID |
| FR03 | User should be able to perform search operation by Name |
| FR04 | User should be able to perform search operation by Description |
| FR05 | User should be able to retrieve a particular recipe by ID |

***Task #2:*** *Create possible* ***ten test cases*** *for created functional requirements, each marked as 3 points.* ***Total: 30 Points***

**ANSWER:**

Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC ID | Description | Steps | Data | Expected Result | Actual Result |
| TC01 | Verify that user cannot create recipe with empty name | User enters empty string for Name and a valid value for Description | Name = ‘ ’  Description = ‘Recipe Info’ | User should get ValidationError, and recipe shouldn’t be in the list | Fail |
| TC02 | Verify that user cannot create recipe with empty description | User enters empty string for Description and a valid value for Name | Name = ‘Recipe1’  Description = ‘ ’ | User should get ValidationError | Fail |
| TC03 | Verify that user can create a new recipe with valid data | User enters valid value for both Name and Description | Name = ‘Recipe1’  Description =  ‘Recipe Info’ | New recipe should appear in recipe list | Pass |
| TC04 | Verify that user cannot update not existing recipe | User tries to update recipe with not existing Id | Id = Not Existing Guid Id | User should receive ‘Not Found’ exception | Fail |
| TC05 | Verify that user can update existing recipe with valid data | User tries to update recipe with existing Id | Id = Existing  Guid Id | Values for the updated recipe should change | Pass |
| TC06 | Verify that user gets empty list for not matching search result (by Name) | User enters not matching Name while searching | Name = ‘Not matching value’ | User should receive empty list | Pass |
| TC07 | Verify that user gets corresponding list of elements for matching search result (by Name) | User enters matching Name while searching | Name = ‘Matching value’ | User should receive list of matching recipes | Pass |
| TC08 | Verify that user can retrieve recipe with existing ID | User tries to retrieve recipe with existing Id value | Id = Existing  Guid Id | User should get existing recipe | Pass |
| TC09 | Verify that user gets corresponding list of elements for matching search result (by Description) | User enters matching Description value while searching | Description = ‘Matching value’ | User should receive list of matching recipes | Pass |
| TC10 | Verify that user gets empty list for not matching search result (by Description) | User enters not matching Description value while searching | Description = ‘Not matching value’ | User should receive empty list | Pass |

***Task #3:*** *Create traceability matrix based on the above test cases and functional requirements. The traceability matrix must contain at least five rows, each marked as 1 point.* ***Total: 5 points***

**ANSWER:**

|  |  |
| --- | --- |
| FR ID | TC ID |
| FR01 | TC01 |
| FR01 | TC02 |
| FR01 | TC03 |
| FR02 | TC04 |
| FR02 | TC05 |
| FR03 | TC06 |
| FR03 | TC07 |
| FR05 | TC08 |
| FR04 | TC09 |
| FR04 | TC10 |

***Task #4****: Create* ***ten MSTEST/NUNIT*** *unit tests for attached code based on your test cases, each marked as 3 points.* ***Total: 30 points***

***Task #5:*** *Use Moq Nuget.Package to Mock Actions log for actions such as Add, Update and Delete, use the necessary constructor for that. Create at least three different Unit Test with the usage of Mock<ActionLogger> to Verify each action method with invocations objects such as Times.Once or Never. (5 points each)*

***Total: 15 Points***

***Task#6:*** *In this task you are required to automate the Yahoo.com search with keyword “**WIUT”. After updating the following code, the browser of your choice/version installed in a PC lab must be automatically started (1), browses into yahoo.com (2), puts the keywords in input box (3) and launches the search (4). At last (5), you must display the first found result title in VS console. Each working correct step updated in the below code is worth 2 points.*

***Total: 10 Points***