**Testing for VR Game Cooktastrophe**

**Aisha – X21341006**

**Black Box Testing - Functionality of Fridge, Functionality of Bin – With Technique Decision Table Testing**

**Fridge Appliance Decision Table**

|  |
| --- |
| Can user open fridge? |
| Can user take items from fridge? |
| Can fridge respawn items? |
| Use fridge? |

3 conditions

1 action

4 rows in the decision table

Determining number of columns:

Can user open fridge – 2 possible values

Can user take items from fridge – 2 possible values

Can fridge respawn items – 2 possible values

Total combinations = 2\*2\*2 = 8 columns

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Can user open fridge? | F | F | F | F | T | T | T | T |
| Can user take items from fridge? | F | F | T | T | F | F | T | T |
| Can fridge respawn items? | F | T | F | T | F | T | F | T |
| Use fridge? | N | N | N | N | N | N | N | Y |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Can user open fridge? | F | F | F | T | T | T | T |
| Can user take items from fridge? | F | T | T | F | F | T | T |
| Can fridge respawn items? | - | F | T | F | T | F | T |
| Use fridge? | N | N | N | N | N | N | Y |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Can user open fridge? | F | F | T | T | T | T |
| Can user take items from fridge? | F | T | F | F | T | T |
| Can fridge respawn items? | - | - | F | T | F | T |
| Use fridge? | N | N | N | N | N | Y |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Can user open fridge? | F | F | T | T | T |
| Can user take items from fridge? | F | T | F | T | T |
| Can fridge respawn items? | - | - | - | F | T |
| Use fridge? | N | N | N | N | Y |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Can user open fridge? | F | T | T | T |
| Can user take items from fridge? | - | F | T | T |
| Can fridge respawn items? | - | - | F | T |
| Use fridge? | N | N | N | Y |

Decision table simplified, outcome with all values as true = Fridge may be used.

4/4 = 1 = 100% code coverage

**Bin Decision Table**

|  |
| --- |
| Can the user throw items in the bin? |
| Will the bin destroy the items ? |
| Will the bin be ready for a new item? |
| Use bin? |

3 conditions

1 action

4 rows in the decision table

Determining number of columns:

Can the user throw items in the bin – 2 possible values

Will the bin destroy the items – 2 possible values

Will the bin be ready for a new item – 2 possible values

Total combinations = 2\*2\*2 = 8 columns

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Can the user throw items in the bin? | F | F | F | F | T | T | T | T |
| Will the bin destroy the items? | F | F | T | T | F | F | T | T |
| Will the bin be ready for a new item? | F | T | F | T | F | T | F | T |
| Use bin? | N | N | N | N | N | N | N | Y |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Can the user throw items in the bin? | F | F | F | T | T | T | T |
| Will the bin destroy the items? | F | T | T | F | F | T | T |
| Will the bin be ready for a new item? | - | F | T | F | T | F | T |
| Use bin? | N | N | N | N | N | N | Y |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Can the user throw items in the bin? | F | F | T | T | T | T |
| Will the bin destroy the items? | F | T | F | F | T | T |
| Will the bin be ready for a new item? | - | - | F | T | F | T |
| Use bin? | N | N | N | N | N | Y |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Can the user throw items in the bin? | F | F | T | T | T |
| Will the bin destroy the items? | F | T | F | T | T |
| Will the bin be ready for a new item? | - | - | - | F | T |
| Use bin? | N | N | N | N | Y |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Can the user throw items in the bin? | F | T | T | T |
| Will the bin destroy the items? | - | F | T | T |
| Will the bin be ready for a new item? | - | - | F | T |
| Use bin? | N | N | N | Y |

Decision table simplified, outcome with all values as true = Bin may be used.

4/4 = 1 = 100% code coverage

**White Box Testing - Functionality of Bin – With Technique Statement Coverage, Branch Coverage**

A screenshot of a computer

Description automatically generated

1 public class FridgeApplianceTest

2 {

3 IEnumerator PrepareScene()

4 {

5 if (sceneIsLoaded)

6 {

7 yield break;

8 }

9 SceneManager.LoadScene(1);

10 yield return null;

11 yield return null;

12 yield return null;

13 sceneIsLoaded = true;

14 }

15 }

In this case there is only really one path for the code to take otherwise it would result in the crash or break of the program as there is no else with this particular if statement. This results in 100% of the codes statement coverage.

Diagram

Description automatically generated