**Coffeemaker Quest Test Plan**

**By Team Maverick**

# **Introduction**

This is the first testing project and we have no experience in testing before. A lot of the test cases that were performed had limitations and drawbacks. It was hard to assess the feasibility of the tests performed. Most of the test processes were carried out on a single device in predefined and contained environment.  Some of the errors that we ran into were event reported acknowledged as errors (for ex. Item Inventory count description and using S to access South doors at the start that kept returning the user to the start of the process)  regarding the test process and development of them we conceptualized them given all the input parameters and output variables that we perceived as valid. Had we been given more time to structure the test report and after to the requirements we would have gone through various testing cycles to try and refine the overall quality.

# Requirements

1. FUN-ITERATION - At each iteration of the game, the user shall be able enter one of six commands - "N" to go North, "S" to go South, "L" to Look for items, "I" for Inventory, "H" for Help, or "D" to Drink. **[There is no “H” for help option]**
2. FUN-UNKNOWN-COMMAND - If a player enters a command not specified by FUN-ITERATION, the system shall respond with the phrase "What?". **[No issues]**
3. FUN-INPUT-CAPS - The system shall be case-insensitive regarding input values; that is, it shall accept capital and lower-case letters and treat them as equivalent. **[Does not take a lower-case “n”, however, accepts all other lower-case letter options]**
4. FUN-MOVE - The system shall allow a player to move North only if a door exists going North, and South only if a door exists going South. **[The system still allows the player to move North or South, even though, the door doesn’t exist on the North or South]**
5. FUN-WIN - The player shall win the game if and only if Coffee, Sugar, and Cream have been collected by the player and then drunk. **[No issues]**
6. FUN-LOSE - The player shall lose the game if and only if the player Drinks but has not collected all the items (Coffee, Sugar, and Cream). **[No issues]**
7. FUN-INVENTORY - Upon entering "I" for inventory, the player shall be informed of the items that he/she has collected (consisting of Coffee, Sugar, and Cream). **[No issues]**
8. FUN-LOOK - Upon entering "L" for Look, the player shall collect any items in the room and those items will be added to the player's inventory. **[The system allows a player to collect the same item unlimited times]**
9. FUN-HELP - Upon entering "H" for Help, the player shall be shown a listing of possible commands and what their effects are. **[There is no “H” for help option]**
10. FUN-UNIQ-ROOM - Each room in the house shall have a unique adjective describing it. **[No issues]**
11. FUN-UNIQ-ROOM-FURNISHING - Each room in the house shall have one and only one unique furnishing visible to the user upon entering the room. **[No issues]**

# Test Cases

**Test Case 1: Check the functionality of each option (UPPER-CASE)**

**Identifier:** Upper\_Case\_value\_Test

**Description:** Test to see the upper-case input values are all functioning.

**Preconditions:** Running the “coffeemaker.jar” file

**Input Values:** N, S,L,I,D

**Execution Steps:** Type each letter option

**Output Values:** A unique value for each input value

**Postconditions:** Proceeding to next step

**Test Case 2: Check the results of each option (LOWER-CASE)**

**Identifier:** Test\_Lower\_case

**Description:** Test to seelower-case input values are all functioning.

**Preconditions:** Running the “coffeemaker.jar” file

**Input Values:** n,s,l,i,d

**Execution Steps:** Type each letter option

**Output Values:** When entered letter “n” the output is **what?**

**Postconditions:** Proceeding to next step

**Test Case 3: Check what is the result if option “D” is chosen before getting the supplies.**

**Identifier:** Drink\_before\_finding

**Description:** Test to see the result if drank before finding items

**Preconditions:** Running the “coffeemaker.jar” file, shouldn’t have any items in the inventory

**Input Values:** D/d

**Execution Steps:** Pressing the D/d button

**Output Values:** You lose!

**Postconditions:** Game ends

**Test Case 4: Check what happens when a player proceeds onwards when no door exists.**

**Identifier: Testing\_nodoor**

**Description:** The player shouldn’t be able to proceed forward if there is no door

**Preconditions:** Running the “coffeemaker.jar” file

**Input Values: S, N,N,N,N,N**

**Execution Steps:** Pressing the “s” buttonin the beginning of the game or N in the rough room.

**Output Values:** Keeps returning to the game beginning (small room).

**Postconditions:** You keep pressing “s”

**Test Case 5: Check what is the result when any one supply is collected and drank.**

**Identifier: Test\_1items**

**Description:** Testing the result when one item (cream) is collected and drank

**Preconditions:** At the start you must go north, and you need to find any two items

**Input Values:** N,S,L,D

**Execution Steps:** In the small room press “L” button

**Output Values:** You Lose!

**Postconditions:** Game exits

**Test Case 6.1: Check what is the result when any two supply items are collected and drunk.**

**Identifier:**  Test\_2items

**Description:** Testing the result when two items are collected and drank

**Preconditions:** At the start you must go, and you need to find any two items

**Input Values:** N,L,N,L,N,L,N,L,D

**Execution Steps:** keep proceeding until to items (coffee and cream) are collected and finally drank.

**Output Values:** you lose with error code 1

**Postconditions:** Game exits

**Test Case 6.2: Check what is the result when any two supply items are collected and drunk.**

**Identifier:**  Test\_2items

**Description:** Testing the result when two items are collected and drank

**Preconditions:** At the start you must go, and you need to find any two items

**Input Values:** N,S,L,N,S,D

**Execution** Steps: keep proceeding north until to two items (cream and cream) are collected and finally drank.

**Output Values:** you lose with error code 1

**Postconditions:** Game exits

**Test Case 6.3: Check what is the result when any two supply items are collected and drunk.**

**Identifier:**  Test\_2items

**Description:** Testing the result when two items are collected and drank

**Preconditions:** At the start you must go north, and you need to find any two items

**Input Values:** N,S,L,N,L,N,L,D

**Execution** Steps: keep proceeding north until two items (cream and coffee) are collected and finally drank.

**Output Values:** Displays you lose with error code 1

**Postconditions:** Game exits

**Test Case 6.4: Check what is the result when any two supply items are collected and drunk.**

**Identifier:**  Test\_2items

**Description:** Testing the result when two items are collected and drank

**Preconditions:** At the start you must go north, and you need to find any two items

**Input Values:** N,L,N,L,N,L,N,L,D

**Execution** Steps: keep proceeding north until to items (coffee & sugar) are collected and finally drank.

**Output Values:** you lose with error code 1

**Postconditions:** Game exits

**Test Case 7: Check how many items are present in the inventory.**

**Identifier: Inventory\_Check**

**Description: The player should be able to check what he has or hasn’t collected**

**Preconditions:** Running the “coffeemaker.jar” file

**Input Values: I, N,L,I,N,L,I,N,L,I**

**Execution Steps:** Press “I” in the beginning, then as you go through the rooms collect and check Inventory

**Output Values:** Shows what is in the inventory

**Postconditions:** Continue to next move

**Test Case 8: Test what is the result when drank after all collecting all three supplies.**

**Identifier: The\_3items\_test**

**Description: Collect all three items and press “D/d”**

**Preconditions: Should have collected all three items**

**Input Values: L, N, L, N,L,N,L,N,L,N,L,N,L,D**

**Execution Steps:** Go through the rooms and collect items, then drink

**Output Values:** You Win!

**Postconditions:** Game Exits

**Test Case 9: Check if upon entering different rooms they have one unique adjective describing them.**

**Identifier:** The\_room\_adj

**Description:** Each room should have an adjective describing them

**Preconditions:** Start the coffeemaker game and go through the rooms

**Input Values:** N,N,N,N,N,N

**Execution Steps:** Go through all rooms

**Output Values:** An adjective describing the room

**Postconditions:** Proceed to next move

**Test Case 10: Check if upon entering different rooms they have only one unique furnishing visible.**

**Identifier:** The \_furnishing\_count

**Description:** Each room should have only one unique furnishing

**Preconditions:** Start the coffeemaker game and go through the rooms

**Input Values:** N,N,N,N,N,N

**Execution Steps:** Go through all rooms

**Output Values:** Only one unique furnishing per room

**Postconditions:** Proceed to next move

**Test Case 11: Check if an item can be collected only once**

**Identifier:** Item\_pickup\_times

**Description:** Player should only be able to pick up an item one time

**Preconditions:** Find an item in the first or any room

**Input Values: L,N,S,L,N,S,L,N,S,L**

**Execution Steps:** Find an item (cream) in small room, collect and move to different room and then return and pick up the same item again. Do this process multiple times

**Output Values:** You found some creamy cream!

**Postconditions:** Proceed to next move or check Inventory

# Traceability Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Requirement 1 | Requirement  2 | Requirement  3 | Requirement  4 | Requirement  5 | Requirement  6 | Requirement  7 | Requirement  8 | Requirement  9 | Requirement  10 | Requirement  11 |
| Test 1 | **X** |  |  |  |  |  |  | **X** | **X** |  |  |
| Test 2 | **X** | **X** | **X** |  |  |  |  | **X** | **X** |  |  |
| Test 3 |  |  |  |  |  | **X** |  |  |  |  |  |
| Test 4 |  |  |  | **X** |  |  |  |  |  |  |  |
| Test 5 |  |  |  |  |  | **X** |  |  |  |  |  |
| Test 6.1 |  |  |  |  |  | **X** |  |  |  |  |  |
| Test 6.2 |  |  |  |  |  | **X** |  |  |  |  |  |
| Test 6.3 |  |  |  |  |  | **X** |  |  |  |  |  |
| Test 6.4 |  |  |  |  |  | **X** |  |  |  |  |  |
| Test 7 |  |  |  |  |  |  | **X** |  |  |  |  |
| Test 8 |  |  |  |  | **X** |  |  |  |  |  |  |
| Test 9 |  |  |  |  |  |  |  |  |  | **X** |  |
| Test 10 |  |  |  |  |  |  |  |  |  |  | **X** |
| Test 11 |  |  |  |  |  |  |  | **X** |  |  |  |

# Defects Found

**Deflect 1:** The “**H”**- help requirement doesn’t exist.

**Identifier: The\_H\_bug**

**Description: There is no “H” for help functionality**

**Reproduction Steps: Pressing “H” or “h” and enter**

**Expected Behavior: That the instructions will include an option “H” for help.**

**Observed Behavior: The “H” for help is non-existent.**

**Deflect 2:** Does not take a lower-case “**n**”, however, accepts all other lower-case letter options.

**Identifier: The\_lowercase\_N\_bug**

**Description: The system doesn’t take lower case “n” command, however, the other lowercase options work.**

**Reproduction Steps: Pressing “n” and enter**

**Expected Behavior: Player should move north**

**Observed Behavior: The system doesn’t recognize the command**

**Deflect 3:** The system **still allows** the player to move north or south, even though, the door doesn’t exist on the north or south.

**Identifier: The\_no\_door**

**Description: Players are still able to proceed forward even though there is no door ahead**

**Reproduction Steps: Pressing “S” in the small room and pressing “N” in the rough room**

**Expected Behavior: The player shouldn’t be able to move forward if there is no door ahead**

**Observed Behavior: The player can move ahead even though there is no door**

**Deflect 4:** The system allows a player to collect the same item **unlimited times.**

**Identifier: The\_unlimited\_collection**

**Description: Players can collect the same item multiple times**

**Reproduction Steps: Pressing the L command to pick up the item**

**Expected Behavior: The item should only be collected once and should display; you already got that specific item when tried to pick up again**

**Observed Behavior: The items can be picked up multiple times**

# Who Did What?

**Rishabh Yata:** Document assembly/design, bug/defect testing, test cases and traceability matrix.

**Sherwin Bothello:** Introduction, testing test cases, overall test plan review.

**Abdulrahman Alrassi:** Testing test cases, verifying defects found, overall test plan review.