**Test Documentation**

**Test Strategy:** Use JUnit to test the functionality of basic operations throughout UserInterface and ItemSearch classes.

**Test Data:** Before every test, UserInterface ui is logged in with Jake’s account ([jake@jakeallen.com](mailto:jake@jakeallen.com), password), and in almost every test a new list is added to the user’s list storage (with names such as “TestList”). Any newly created lists are removed from the ListStorage, so every test can operate on a clean slate with the user’s list storage. Also, for item-related use cases, a custom item “Nintendo Wii U Console” from Store “Amazon” with quantity 3 is used.

**Test Cases:**

* addAndDeleteList() – adds list “TestAdd” to the user’s ListStorage; if the adding fails, or a separate search through the ListStorage reveals the absence of the list, the test fails. A sub-function deleteList() is called to test deleting this same list from ListStorage, failing if the list is still there. deleteList() is not a separate test because JUnit calls tests randomly, and deleteList() depends on an existing list

These functions both add a list to start and delete it to finish:

* copyList() – copies the list “TestList” as a new list “CopyList” – failing if the copy doesn’t work, or the copied list is not in the ListStorage
* addItem() – adds an item to “TestList”, failing if the added item is not present
* deleteItem() – adds and deletes item from “TestList”, failing if the item is still in the list
* crossItem() – crosses off the newly added item, failing if the item’s quantity is not negative (as any crossed item should have their quantity be)
* uncrossItem() – uncrosses the newly crossed-off item, failing if the item’s quantity is negative (a non-crossed item always has positive quantity)

These last functions do not involve adding new lists:

* storeSelect() – initializes an ItemSearch object and has it switch to a table of the first store – while Swing is not normally visible to the user in JUnit, this test calls isTableVisible() on the ItemSearch, which reports if the table would otherwise be visible in a normally run program. The test fails if this returns false
* itemSearch() – the ItemSearch object’s filters are set to only set items of type “Bread” & extra “wheat” to be the only items visible in the table; if "Bread Factory Bread wheat" is not the first item, the test fails

**Traceability Matrix:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Requirements | Add List | Copy List | Delete List | Add Item | Select Store | Set Filter | Cross Item | Uncross Item | Delete Item |
| TC 1: addAndDeleteList() | ✓ |  | ✓ |  |  |  |  |  |  |
| TC 2: copyList() | ✓ | ✓ | ✓ |  |  |  |  |  |  |
| TC 3: addItem() | ✓ |  | ✓ | ✓ |  |  |  |  |  |
| TC 4: deleteItem() | ✓ |  | ✓ | ✓ |  |  |  |  | ✓ |
| TC 5: crossItem() | ✓ |  | ✓ | ✓ |  |  | ✓ |  |  |
| TC 6: uncrossItem() | ✓ |  | ✓ | ✓ |  |  |  | ✓ |  |
| TC 7: storeSelect() |  |  |  |  | ✓ |  |  |  |  |
| TC 8 itemSearch() |  |  |  |  |  | Incomplete |  |  |  |