Project 4 System Testing

Test Case	Requirement	Test Description/Input Data	Expected Result/Output
1	Won’t Crash when trying to add coffee to basket with not enough input.	Press ‘Add to Basket’ with all empty fields.	Alert with an error message telling the user to fill out the required fields.
2	We can clear selections from the coffee view to essentially start our coffee item creation over.	Fill out a bunch of the fields in the coffee view, then press the ‘clear the order’ button.	All items were unselected as we wanted.
3	We can add coffee(s) to the basket.	Add multiple different coffee items to the basket, then check the basket view to see if they are there.	The coffee items that we added were indeed in the basket.
4	We must be able to return to the main view from any view in the app.	Continually switch views by going into any of the non-main views, then use the ‘return home’ or the ‘return to ru café home’ button.	We are able to seamlessly switch between the 5 views.
5	Won’t crash when trying to add a donut to it’s listview without filling out the required fields.	Continuously press the add ‘>>>’ button in order donuts view to try to cause an error.	An error alert was given to the user each time the add ‘>>>’ button was pressed. Thus, the error was properly handled.
6	Won’t crash when trying to remove a donut from the listview in the donut view without actually selecting an item to remove.	We will continue to press the remove ‘<<<’ button.	An error alert was given to the user, properly handling the misuse of the ‘<<<’ remove item from list view button.
7	When selecting donut type from the drop down, the available flavors for that type are listed in the listview just below the drop down. Additionally, the	We continuously changed the donut type, making sure each time that the proper donut flavors were listed and the correct image was shown.	The image and flavor options were displayed properly.

	corresponding image must change each time the donut type is changed.		
8	We must be able to add and remove donuts to the listview in the donut view scene.	We added various numbers of donuts to the list, removed some, added more, and removed all of them.	The add '>>>' and remove '<<<' features worked seamlessly.
9	We must be able to add a set of varying donuts to the basket.	Add varying number of donuts to our running listview, then add them to the basket.	In each case, we successfully added the donuts that we wanted to the basket.
10	We must be able to keep running subtotals (and count the tax and total price) in various windows.	**Coffee:** 2 venti with 2 addins **Donuts:** 5 cake donuts (Various other tests were run, this is just a short example for the sake of keeping this concise).	**Coffee Subtotal:** 6.78 **Donuts Subtotal:** 8.95 **Total Subtotal:** 15.73 **Tax:** 1.04 **Total:** 16.77
11	Must be able to remove a selected item from the current basket.	Add various items to the basket. Press the 'Remove Selected Item'.	We were printing lists to the console to confirm that the items were in fact being removed from the basket.
12	Must be able to place an order with the given items in the basket.	Add varying numbers of items to the basket, then place an order.	All the items, along with the total and an order number were sent to an order object that we created.
13	Must be able to display all store orders.	Add varying store orders to the basket, then place the order. We tried adding an order with just coffees, and order with just donuts, and an order with mixed items.	All of the orders were printed into the listview as expected.
14	We must be able to cancel an order.	Create many orders, then continue to cancel them, making sure they are not selectable from the drop down, and	Orders were cancelled as expected. We were able to print orders to the console as a way of checking that they

		continue to print out the remaining orders. We can repeat this process until all orders have been cancelled.	were also removed in the backend.
15	We must be able to export orders to the text file defined in the store orders controller class (StoreOrders.txt), which is stored in the main project folder.	Place various orders, then click the ‘export orders’ button.	We chose to truncate the file each time it was opened, therefore whenever we export orders, we have a better updated order list. This was useful in dealing with cancelled orders or possibly preventing repeat orders (if we had alternatively chose to append the text or something similar).