**Group 07 Test Plan**

Konrad Kapusta and Triet Lieu

SDSU Spring 2024: CS 250

\*\*submit as pdf when we’re done

**UNIT TESTING**

SET 1: Transactions

| Testing total cost method | Add items with prices $10 and $5 to cart. Call getTotalCost(). | Expected output: $15. |
| --- | --- | --- |
| Testing making a sale | Add item with ID number 20 to cart. getAllItems() under ItemType tells us there are 5 of this item. Call makeSales() on the item in cart. | Expected output: getAllItems() returns 4. |
| Testing card payment | 1. Add valid Card card into transaction. Call verifyPayment(card). 2. Add invalid Card card into transaction. Call verifyPayment(card). | 1. Expected output: True. 2. Expected output: False. |
| Testing cash payment | 1. getTotalCost() returns $15. Call payCash(16). 2. getTotalCost() returns $15. Call payCash(14). | 1. Expected output: True. 2. Expected output: False. |

SET 2: Transaction History

| Testing adding transaction to history | Initialize accessesTransactionHistory as an empty list.  Use addTransaction(Transaction\_1)  Through Manager class, use Manager\_1.accessesTransactionHistory. | Expected output: a list with just the Transaction\_1 object |
| --- | --- | --- |
| Testing removing transaction from history | Now, use removeTransaction(Transaction\_1)  Use Manager\_1.accessesTransactionHistory. | Expected output: an empty list |
| Testing removing nonexistent transaction to history | removeTransaction(Transaction\_2) | Expected output: Error, cannot remove transaction that does not exist. |

SET 3: Items

| Test item ID | setID for item\_1 to be 999 | Expected output:  item\_1.ID returns 999 |
| --- | --- | --- |
| Test item price | setPrice for item\_1 to be 999 | Expected output: item\_1.price returns 999 |
| Test selling an item | item\_1.makeSale() | Expected output: True  ItemType.getAllItems(item\_1) has decreased by 1. |
| Test date created for items | setDateCreated for item\_1 to be “03162024” | Expected output: item\_1.dateCreated returns “03162024” |

SET 4: Cloud Server

| Testing backup frequency | 1. Set backup to hourly 2. Set backup to daily 3. Set backup to weekly | Expected output:   1. Successful backup done each hour 2. Successful backup done once a day 3. Successful backup done once a week |
| --- | --- | --- |
| Testing encryption settings | 1. Disable encryption in settings 2. Enable encryption in settings | Expected output:   1. Server settings display encryption as OFF 2. Server settings display encryption as ON |

**SYSTEM TESTING**

SET 1: Cloud service server/backup

| Testing data backup | Call backup manually | Expected output: Successful backup notification and backup data matches actual system data |
| --- | --- | --- |
| Testing data synchronization | Update inventory for item 1 in store ID 1 by setting items attribute for ItemType to 100. | Expected output:  Checking that in store with ID 2. The items count for item 1 is 100. |

**INTEGRATION TESTING**

SET 1: Credit Card Processing

| Testing makeSales() in transaction with credit cards | Call verifyPayment() on Test\_Card (valid) before the transaction is confirmed. | Expected output: Card class calls checkValidity(Test\_Card) and the credit Card API returns true. Transaction processes. |
| --- | --- | --- |
|  |  |  |