|  |  |  |
| --- | --- | --- |
| Test Scenario | Step in Flow | Alternative |
| Scenario 1 - Successful Sale | Basis |  |
| Scenario 2 - Customer non-existent | Basis | 1a |
| Scenario 3 - Product non-existent | Basis | 3a |
| Scenario 4 – Invalid quantity | Basis | 3b |

The table above is in relation to the fully dressed use case of createSale. It is used to formulate possible scenarios which can be analysed further as shown in the table below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Scenario | Input | | | Expected Result |
| Customer Name | Product Name | Quantity |
| Q 1 | Successful Sale | V | V | V | * Customer is found in the system. * Product is found in the system. * Legitimate quantity is inserted. * A sale is made |
| Q 2 | Customer doesn’t exist | I | V | V | A message should be sent to the user that there is no such customer. |
| Q 3 | Product doesn’t exist | V | I | V | A message should be sent to the user that there is no such product. |
| Q 4 | Bad quantity input | V | V | I | A message should be sent to the user that there is not enough items for the order. |

This table represents a black box test. It focuses solely on input and output, ignoring the internal structure and implementation. The advantage is that the test can be reused as it is, regardless of future changes and improvements in the code.

Another important thing to test are input ranges. They are best analysed by using equivalence class partitioning. This is done by identifying valid classes and invalid. Each legal input is tested just below and above the set boundary.

0

Amount in Stock

Invalid Valid

0 Amount in Stock

Amount

Invalid Valid Invalid

This shows that an item can only be sold if there is an amount in stock that is greater than zero. If this is true, the system should allow the sale to proceed. If not, a message should be sent to the user that there are not enough items in stock.

Below are test cases defined with basis in equivalence classes and boundary values.

|  |  |  |  |
| --- | --- | --- | --- |
| Method: public int AddSaleLine(name, quantity) | | | |
| Test Case No. | Initialisation State  item.getStock() | Input  Amount | Expected Output |
| 1 | 20 | 1 | Return 1  In stock: 19 |
| 2 | 20 | 20 | Return 20  In stock: 0 |
| 3 | 20 | 0 | Return 0  In stock: 20 |
| 4 | 20 | 21 | Return 0  In stock: 20 |
| 5 | 20 | -1 | Return 0  In stock: 20 |
| 6 | 0 | 1 | Return 0  In stock: 0 |

For a customer to accept the system, it is important that the requirements are measurable. That also helps the developer to know when they are met. It is achieved by using measurability criteria.

Acceptance Criteria for Create Sale:

* Pre-conditions: Employee is logged in, Customer and Item are registered in the system
* Must hold the information: saleNr, discount, dateCreated, isPacked, datePacked, isSent, dateSent, isPaid, datePaid, employee, customer, saleLines
* saleNr, discount, dateCreated, isPaid employee, customer and saleLines have to be filled in as a minimum
* saleNr can not no more than 32 characters.

Quality requirements can be classified into five main categories: features, usability reliability, performance, supportability(Larman ch.5, p. 57). In short it is called FURPS. However, not all of them can be implemented with limited time and resources. Below are the ones that were chosen for this specific project.

Quality criteria:

1. Robustness - the IT system should be able to handle multi users without complains, evading deadlocks. If a failure occurs, it should restart in less than a minute. Also it shouldn’t corrupt any data. (Changes made should be abandoned if the action wasn’t finished.)
2. Reliability - the percentage of uptime should be around 98%. If the database is not working, the system should try to reconnect every minute.
3. Ease of Use - employees should take no more than two hours to learn the new software. (Each type of employee (Warehouse worker, cashier and manage have different access to the system which makes their use much simpler and the user interface not overcrowded with unneeded information.)
4. Security - password protection for all three types of employees.