COMPLIANCE REPORT

for

Zulu - A Motor Part Shop Software

Prepared by -

Ashutosh Kumar Singh (19CS30008)

Vanshita Garg (19CS10064)

Suhas Jain (19CS30048)

Indian Institute of Technology, Kharagpur

April 5, 2021

Contents

1 Introduction and Important Points

This is the Compliance Report document for Zulu - a Motor Part Shop Software. Here, we list down test cases that were mentioned in the test suite along with the obtained PASS/FAIL results that were obtained after testing the software. The exact test case and the golden outputs can be checked under the same corresponding sections in the test suite document.

Note : At all places in this document, the numbers written in parentheses after the headings of a section or subsection denote the corresponding section number in the Test Plan Document.
Also, note that for a positive test case, the expected result should be **PASSED**, while for a negative test case, the expected result should be **FAILED**.

2 Compliance Report for the Frontend GUI Interface

2.1 Home Page (5.1)

Working Properly

2.2 Login Page (5.2)

1. *Both Username and Password are correct*

   PASSED

2. *Username is correct but Password is incorrect*

   FAILED

3. *Username is incorrect but Password is correct*

   FAILED

4. *Both Username and Password are incorrect*

   FAILED

2.3 Dashboard (5.3)

1. *Working of all buttons*

   Working Properly

2.4 Add an Item (5.4)

1. *Working of all drop-down menus*

   PASSED

2. *Working of all text fields*

   PASSED

3. *Working of all buttons*

   PASSED

4. *The data entered in all the fields is valid / correct*

   PASSED

5. *Item Type entered is invalid*

   FAILED

6. *Manufacturer Name entered is invalid*

   FAILED

7. *Manufacturer Address is invalid*

   FAILED

8. *Vehicle Type entered is invalid*

   FAILED

9. *Price entered is not a number*

   FAILED

10. *Price of the item entered is zero*

    FAILED

11. *Price of the item entered is negative*

    FAILED

12. *Initial Quantity entered is not a number*

    FAILED

13. *Initial Quantity entered is zero*

    FAILED

14. *Initial Quantity entered is negative*

    FAILED

2.5 Remove an Item (5.5)

1. Working of drop down menus

   PASSED

2. *Working of all buttons*

   PASSED

3. *All the fields chosen - Item Type, Manufacturer Name and Vehicle Type are valid*

   PASSED

2.6 Report a Sale (5.6)

1. *Working of all drop-down menus*

   PASSED

2. *Working of all text fields*

   PASSED

3. *Working of all buttons*

   PASSED

4. *All the fields chosen and data entered are valid*

   PASSED

5. *Quantity entered is not a number*

   FAILED

6. *Quantity entered is zero*

   FAILED

7. *Quantity entered is negative*

   FAILED

8. *Quantity entered is greater than the quantity in the inventory*

   FAILED

2.7 View Inventory (5.7)

1. *Working of the scrollable list of items*

   PASSED

2. *Working of all buttons*

   PASSED

2.8 Day - End Tasks (5.8)

1. *Computation of number of items to be ordered at the end of a day*

   PASSED

2. *Working of the generated order list, which will be a scrollable list*

   PASSED

3. *Working of all buttons*

   PASSED

2.9 Graph for Daily Sales of a Month (5.9)

1. *View the graph before the first month has ended*

   FAILED

2. *View the graph on the day a month has ended*

   PASSED

3. *View the graph in the middle of a month*

   PASSED

3 Compliance Report for Backend Classes and Database Management

3.1 Testing the `Owner` class (6.1)

3.1.1 Testing the `getName()` function (6.1.1)

1. *Retrieve and verify the* `name` *of the owner*

   PASSED

3.1.2 Testing the `setName(String name)` function (6.1.2)

1. *Set the* `name` *of the owner to a valid string*

   PASSED

2. *Set the* `name` *of the owner to an invalid string*

   FAILED

3.1.3 Testing the `getUsername()` function (6.1.3)

1. *Retrieve and verify the* `username` *of the owner*

   PASSED

3.1.4 Testing the `setUsername(String username)` function (6.1.4)

1. *Set the* `username` *of the owner to a valid string*

   PASSED

2. *Set the* `username` *of the owner to an invalid string*

   FAILED

3.1.5 Testing the `validate(String username, String password)` function (6.1.5)

1. *Both the* `username` *and* `password` *passed are the same as the actual* `username` *and* `password` *of the owner*

   PASSED

2. `username` *is correct but* `password` *is incorrect*

   FAILED

3. `username` *is incorrect but* `password` *is correct*

   FAILED

4. *Both* `username` *and* `password` *are incorrect*

   FAILED

3.2 Testing the `Item` class (6.2)

3.2.1 Testing the Constructor `Item(String type, double price, int quantity, int manufacturerID, String vehicleType, Date startDate)` (6.2.1)

Implicitly tested in other sections

3.2.2 Testing the `getUID()` function (6.2.2)

1. *Retrieve and verify the* `uID` *of an* `Item` *object.*

   PASSED

3.2.3 Testing the `getType()` function (6.2.3)

1. *Retrieve and verify the* `type` *of an* `Item` *object.*

   PASSED

3.2.4 Testing the `getPrice()` function (6.2.4)

1. *Retrieve and verify the* `price` *of an* `Item` *object.*

   PASSED

3.2.5 Testing the `getQuantity()` function (6.2.5)

1. *Retrieve and verify the* `quantity` *of an* `Item` *object.*

   PASSED

3.2.6 Testing the `getManufacturerID()` function (6.2.6)

1. *Retrieve and verify the* `manufacturerID` *of an* `Item` *object.*

   PASSED

3.2.7 Testing the `getVehicleType()` function (6.2.7)

1. *Retrieve and verify the* `vehicleType` *of an* `Item` *object.*

   PASSED

3.2.8 Testing the `getStartDate()` function (6.2.8)

1. *Retrieve and verify the* `startDate` *of an* `Item` *object.*

   PASSED

3.2.9 Testing the `save()` function (6.2.9)

1. *Insert a new item to the database when it is empty*

   PASSED

2. *Insert a new item to the database when it is not empty*

   PASSED

3.2.10 Testing the `delete()` function (6.2.10)

1. *Delete a item when multiple items are present in the inventory database*

   PASSED

2. *Delete a item when only a single item is present in the inventory database*

   PASSED

3.2.11 Testing the `updateSale(int numSold)` function (6.2.11)

1. *Update the quantity of an item being sold*

   PASSED

3.3 Testing the `Manufacturer` class (6.3)

3.3.1 Testing the Constructor `Manufacturer(String name, String address)` (6.3.1)

Implicitly tested in other sections

3.3.2 Testing the `getUID()` function (6.3.2)

1. *Retrieve and verify the* `uID` *of a* `Manufacturer` *object.*

   PASSED

3.3.3 Testing the `getName()` function (6.3.3)

1. *Retrieve and verify the* `name` *of a* `Manufacturer` *object.*

   PASSED

3.3.4 Testing the `getAddress()` function (6.3.4)

1. *Retrieve and verify the* `address` *of a* `Manufacturer` *object.*

   PASSED

3.3.5 Testing the `getItemCount()` function (6.3.5)

1. *Retrieve and verify the* `itemCount` *of a* `Manufacturer` *object.*

   PASSED

3.3.6 Testing the `save()` function (6.3.6)

1. *Insert a new manufacturer to the database when it is empty*

   PASSED

2. *Insert a new manufacturer to the database when it is not empty*

   PASSED

3.3.7 Testing the `delete()` function (6.3.7)

1. *Delete a manufacturer when multiple manufacturers are present in the inventory database*

   PASSED

2. *Delete a manufacturer when only a single manufacturer is present in the inventory database*

   PASSED

3.4 Testing the `Inventory` class (6.4)

3.4.1 Testing the `retrieveData()` function (6.4.1)

1. *The inventory database is empty*

   PASSED

2. *The inventory database is not empty*

   PASSED

3.4.2 Testing the `removeItem(int itemID)` function (6.4.2)

1. *Remove an item when multiple items are present in the inventory database*

   PASSED

2. *Remove an item when only a single item is present in the inventory*

   PASSED

3. *The manufacturer of the item being deleted makes some other item(s) too.*

   PASSED

4. *The manufacturer of the item being deleted does not make any other item.*

   PASSED