TEST PLAN

for

A Motor Part Shop Software

Prepared by -

Jatin Gupta (20CS10087)

Gopal (20CS30021)

Jay Kumar Thakur (20CS30024)

Indian Institute of Technology, Kharagpur

March 23, 2022

Contents

1. Introduction 4

2. Test Items 4

3. Software Risk Issues 4

4. Features to be tested 5

5. Features not to be tested 5

6. Approach 5

6.1. Level of Testing

6.2. Test Tools

6.3. Testing the Backend

6.4. Testing the Database

6.5. Testing the Frontend

7. Pass/Fail Criteria 6

8. Test Deliverables 7

9. Environmental Needs 7

10. Test Scenarios for the Frontend GUI Interface 7

10.1. Home Page

10.2. Login Page

10.3. Dashboard

10.4. Modify the item database

10.4.1. Add a new item

10.4.2. Adding a manufacturer to the item

10.5. Modify the manufacturer Database

10.5.1. Add a new manufacturer

10.5.2. Viewing the manufactured items of a manufacturer

10.6. Buy an item

10.7. Sell an item

10.8. View Inventory

10.9. END Day Tasks

10.10. Graph for daily sales of a month

11. Test Scenarios for Backend Classes and Database Management..... 9

11.1. Testing the Owner class

11.1.1. Testing the getUsername() function

11.1.2. Testing the setUsername(String username) function

11.1.3. Testing the getPassword() function

11.1.4. Testing the setPassword(String password) function
11.1.5. Testing the validate(String username, String password) function
11.2. Testing the item class
11.2.1. Testing the constructor
11.2.2. Testing the getPart_ID() function
11.2.3. Testing the getType() function
11.2.4. Testing the getPrice() function
11.2.5. Testing the getQuantity() function
11.2.6. Testing the getVehicleType() function
11.2.7. Testing the getRack_Name() function
11.2.8. Testing the getManufacturer_list() function
11.2.9. Testing the setType(string type) function
11.2.10. Testing the setPrice(Float price) function
11.2.11. Testing the setQuantity(Int quantity) function
11.2.12. Testing the setVehicleType(string type) function
11.2.13. Testing the setRack_Name() function
11.2.14. Testing the save() function
11.2.15. Testing the delete() function
11.2.16. Testing the addManufacturer(Manufacturer a, float cost_price) function
11.2.17. Testing the updateStock(Int quantity) function
11.3. Testing the Manufacturer class
11.3.1. Testing the constructor
11.3.2. Testing the getManufacturerID() function
11.3.3. Testing the getName() function
11.3.4. Testing the getAddress() function
11.3.5. Testing the getPhone_no() function
11.3.6. Testing the getItem_List() function
11.3.7. Testing the setName(string name) function
11.3.8. Testing the setAddress(string address) function
11.3.9. Testing the setPhone_no(Int phoneno) function
11.3.10. Testing the save() function
11.3.11. Testing the delete() function
11.3.12. Testing the AddItem(Item a) function
11.4. Testing the invoice class
11.4.1. Testing the constructor
11.4.2. Testing the addItem() function
11.4.3. Tetsing the getRevenue() function
11.5. Testing the database class
11.5.1. Testing the retrieveItem() function.
11.5.2. Testing the retrieveManufacturer() function
11.5.3. Testing the retrieveInvoice() function
11.5.4. Testing the addItem(Item a) function
11.5.5. Testing the addManufacturer (Manufacturer a) function
11.5.6. Testing the addInvoice(Invoice invoice) function
11.5.7. Testing the deleteItem(Item a) function
11.5.8. Testing the deleteManufacturer(Manufacturer a) function
12. References 14

1. Introduction

This is the Master Test Plan for our Motor Part Shop Software. This document mainly addresses the various features that will be tested. This document provides a brief overview of the features that need to be tested. The primary motive of this plan is to ensure that at the end of the testing process, the software performs all the tasks as expected. It will also prove the reliability of the software so that it never goes to an inconsistent state or crashes unexpectedly.

Presently, only the unit testing is being done, i.e, all the functions and modules will be tested individually. The software is executed for various scenarios and then the output produced is cross-checked with the theoretical golden output either manually or automatically.

2. Test Items

The following features are intended to be tested in this test plan:

- Logging into the software
- Adding a new item/part to the inventory
- Adding the stock of the item to the inventory
- Removing a specific number of items from the the item stock inventory
- Viewing the current inventory
- Generating the list of items whose stock is below the threshold value
- Generating the revenue of the sales of a day
- Plotting the graph of the sales for a month
- Removing the item

3. Software Risk Issues

- This software is intended to be used by a single user. Multiple users cannot use this software.
- There are several parts of the software which are not within the scope of this project. We cannot include testing of such parts. There may be certain requirements which haven’t yet been recognized. These requirements can be updated later.
- Database handling is also important to avoid the corruption of the software.
- The ability to restart the software in the middle of a process may lead to a crash of the software.

4. Features to be tested

- Checking all the constructors and method functions of all classes used to implement the software. The getter and setters of the classes are also checked.
- Login of the user (Shop owner) - The shop owner only will be able to login to the user. This ensures that the database and the software data cannot be hampered by any unauthorized user.
- Adding a new item to the database - The new item will be added to the database. The item and its stock will be added to the inventory.
- Adding a manufacturer to the item's manufacturer hashmap - The item class contains a hashmap of manufacturers and cost price. A manufacturer can be added to the item's manufacturer's hashmap.
- Removing a manufacturer from the item's manufacturer hashmap - The given function will be removed from the item's hashmap
- Adding a manufacturer to the database - This function will add a new function to the database.
- Buying an item - To buy the item from a manufacturer if the stock of the item in the inventory is less than the threshold. The owner will select this option to buy the item. The stock of the item in the inventory will be updated after this process.
- Selling an item - To sell an item to the customer. The owner will use this function to notify the software about the sale. The stock of the item in the inventory will be reduced and the sale transaction will be stored in the database to calculate the total revenue at the end of day and to plot the revenues of a month.
- Viewing the inventory of the shop - This function will display the inventory of the shop.
- Notifying the owner about the items whose stock is below the threshold - At the end of day, if the stock of an item in the inventory of the shop is less than the calculated threshold, it will be displayed to inform the user about such items.
- Generating the revenue of each day - At the end of each day, the revenue of the whole day, i.e., the total sales of that day will be displayed.
- Plotting the graph - At the end of month, the owner can see the plot of sales of the shop for that month. The graph contains days in x-axis and the revenue generated on that day in y-axis.

5. Features not to be tested

- The hardware components like the printer and others are not tested.
- Extreme test cases are not used.
- All the third party libraries are not tested.

6. Approach (Strategy)

6.1 Level of Testing

The software will be tested from unit and application tests.

- All the individual units of the software will be tested in order to validate that each unit is working properly. All the constructors, getters, setters and functions of all the classes are tested.
- Application testing is carried out to verify the working of the complete software. It evaluates the end-to-end system specifications and verifies the interactions between the modules of the software.

6.2 Test tools

For performing unit testing, we are using the JUnit 4 testing framework which helps in easing out the testing process.

6.3 Testing the Backend

Here, we test the functional logic of each method and track changes made to the data members of the classes and the states of the objects. We make a testing class for each of the backend classes, which have unit testing methods for each of the individual methods in the classes. The output generated is verified against the golden output automatically here. If there is a mismatch then an appropriate message is displayed.

6.4 Testing the Database

The database queries and updates mostly happen inside the methods of the backend classes simultaneously with the updation of the state of the objects. So, there is no need to write separate methods for testing this part. The tests for verifying that the database is always updated properly can be written in the unit testing functions described in the previous part itself.

6.5 Testing the Frontend

This is the part where we need to verify that the GUI is functioning as expected and everything is positioned on the screen properly. Automating this task is extremely difficult, and hence this has to be done manually. So, here we just manually proceed, entering the appropriate inputs and go on checking if the correct results are displayed in an appropriate format.

7. Pass/Fail Criteria

Unit test level:

- All given test cases are to be completed.
- No minor errors should be detected.

- All parts of the code should be convered.

Application test level

- Unit test should be completed
- Slowly implementing parts of code and checking
- No detected bugs in integrated parts
- After complete integration it should work with minor errors

8. Test Deliverables

- Test Plan Document
- Test cases
- Test Design Specification

9. Environmental Needs

The software uses the JUnit framework for unit testing. The system should have access to the database. Basic hardware is also required for the software to function properly. The folowing test have been done in a JAVA integrated development environment using IntelliJ IDEA 2021.3.2.
The system will be tested in local network.

10. Test Scenarios for the Frontend GUI Interface

10.1 Home Page

- This window shows the welcome page of the software. Nothing has to be tested in this using test cases.

10.2 Login Page

- Both Username and Password are correct [Pass]
- Username is correct but Password is incorrect [Fail]
- Username is incorrect but Password is correct [Fail]
- Both Username and Password are incorrect [Fail]

10.3 Dashboard

- Testing of all buttons [Pass]

10.4 Modify the item database

- Testing of all buttons [Pass]
- Display of the item list [Pass]

10.4.1 Add a new item

- Testing of input string name of the item [Pass]
- The input stock amount is a positive integer [Pass]
- The input stock amount is 0 [Fail]
- The input stock amount is a string [Fail]
- Input vehicle type is a valid string [Pass]
- Input vehicle type is not valid [Fail]
- Input rack is valid [Pass]

10.4.2 Adding a manufacturer to the item

- The user has to select the item whose inventory has to be updated from the list. The product selected has to be modified.
- Checking the Add Button [Pass]
    - The input cost price should be a positive number. [Pass]
    - The input cost price is 0 or negative [Fail]
    - The input cost price is string [Fail]
- Checking the Remove Button [Pass]
- Selecting the manufacturer.
- Checking whether a popup has been displayed if the manufacturer has been added/removed to the item's manufacturer list.

10.5 Modify the manufacturer database

- Testing of all buttons [Pass]
- Display of the manufacturer list [Pass]

10.5.1 Add a new manufacturer

- Testing of the input manufacturer name [Pass]
- Testing of the input address of the manufacturer [Pass]
- Testing of the correctness of contact of the manufacturer [Pass]
- Contact entered is a string [Fail]

10.5.2 Viewing the manufactured items of a manufacturer

- Selecting the manufacturer from the manufacturer list.
- Checking the "View Manufactured items" button [Pass]
- Display of the manufactured items of the manufacturer [Pass]

10.6 Buy an Item

- Checking of all buttons [Pass]
- Display of the entire item list [Pass]
- Selection of the required item [Pass]
- The input given in the quantity field is positive integer [Pass]
- The input given in the quantity field is zero [Fail]
- The input given in the quantity field is negative integer [Fail]
- The input given in the quantity field is string [Fail]

10.7 Sell an Item

- Checking of all buttons [Pass]
- Display of the entire item list [Pass]
- Selection of the required item [Pass]
- The input given in the quantity field is zero [Fail]
- The input given in the quantity field is negative integer [Fail]
- The input given in the quantity field is string [Fail]
- The input given in the quantity field is greater than Stock [Fail]
- The input given in the price field is negative number [Fail]
- The input given in the price field is string [Fail]

10.8 View Inventory

- Working of the scrollable list of items [Pass]

10.9 END Day Tasks

- Display of Revenue for the day [Pass]
- Working of the generated order list [Pass]

10.10 Graph for Daily Sales of a Month

- View the graph on the day a month has ended [Pass]
- View the graph before the first month has ended [Fail]
- View the graph in the middle of a month [Pass]

11. Test Scenarios for backend classes and database management

11.1 Testing the Owner class

11.1.1 Testing the getUsername() function

- Retrieve and verify the name of the owner [Pass]

11.1.2 Testing the setUsername(String username) function

- Set the username of the owner to a valid string [Pass]
- Set the username of the owner to an invalid string [Fail]

11.1.3 Testing the getPassword() function

- Retrieve and verify the password of the owner [Pass]

11.1.4 Testing the setPassword(String password) function

- Set the password of the owner to a valid string (length of the string is more than 4)[Pass]
- Set the password of the owner to an invalid string (length of the string is less than 5 characters) [Fail]

11.1.5 Testing the validate(String username, String password) function

- Both the username and password passed are the same as the actual username and password of the owner [Pass]
- username is correct but password is incorrect [Fail]
- username is incorrect but password is correct [Fail]
- Both username and password are incorrect [Fail]

11.2 Testing the Item class

11.2.1 Testing the Constructor Item(String name, String type, double price, int quantity, String vehicleType, String rack)

The constructor is called only after ensuring that all the parameters passed are valid. So, they will be tested either in the GUI testing or in the database testing. So, there is no need to test the constructor here.

11.2.2 Testing the getPart_ID() function

- Retrieve and verify the uID of an Item object. [Pass]

11.2.3 Testing the getType() function

- Retrieve and verify the type of an Item object. [Pass]

11.2.4 Testing the getPrice() function

- Retrieve and verify the price of an Item object. [Pass]

11.2.5 Testing the getQuantity() function

- Retrieve and verify the quantity of an Item object. [Pass]

11.2.6 Testing the getVehicleType() function

- Retrieve and verify the vehicleType of an Item object. [Pass]

11.2.7 Testing the getRack_Name() function

- Retrieve and verify the Rack of the item object. [Pass]

11.2.8 Testing the getManufacturer_list() function

- Retrieve the item object's manufacturer's list (Hashmap) and return it. [Pass]

11.2.9 Testing the setType(string type) function

- Set the type of item to the passed string type. [Pass]

11.2.10 Testing the setPrice(Float price) function

- Checking whether the function sets the price of the item to the price passed to the function. [Pass]

11.2.11 Testing the setQuantity(Int quantity) function

- Checking whether the function sets the stock of the item in the inventor to the quantity passed to the function. [Pass]

11.2.12 Testing the setVehicleType(string type) function

- Checking whether the function sets the vehicle type to the vehicle type passed to the function. [Pass]

11.2.13 Testing the setRack_Name() function

- Checking whether the function sets the Rack to the string rack passed to the function. [Pass]

11.2.14 Testing the save() function

- Insert an item to the database when the item is not in database [Pass]
- Insert an item to the database when the item is in database [Fail]

11.2.15 Testing the delete() function

- Deleting the item present in the inventory database [Pass]
- Not deleting the item if the item is not present in the database. [Fail]

11.2.16 Testing the addManufacturer(Manufacturer a, float cost_price) function

- Adding the manufacturer and the manufacturer's cost price for that item to the manufacturer list (Hashmap) of the item. [Pass]
- Not adding the manufacturer if it's already in the item's manufacturer list [Fail]

- Updating the manufacturer's item list and adding this item object in that list [Pass]

11.2.17 Testing the updateStock(Int quantity) function

- Updating the quantity of the item after the buy and sell operations [Pass]

11.3 Testing the Manufacturer class

11.3.1 Testing the Constructor Manufacturer(String name, String Address, Contact phone)
The constructor is called only after ensuring that all the parameters passed are valid. So, they will be tested either in the GUI testing or in the database testing. So, there is no need to test the constructor here.

11.3.2 Testing the getManufacturerID() function
- Retrieve and verify the Manufacturer ID of a Manufacturer object. [Pass]

11.3.3 Testing the getName() function
- Retrieve and verify the name of a Manufacturer object. [Pass]

11.3.4 Testing the getAddress() function
- Retrieve and verify the address of a Manufacturer object. [Pass]

11.3.5 Testing the getPhone_no() function
- Retrieve and verify the phone number of the manufacturer [Pass]

11.3.6 Testing the getItem_List() function
- Retrieve the Item List of the manufacturer item and return it [Pass]

11.3.7 Testing the setName(string name) function
- Checking whether the name of the manufacturer is set to the string name passed to the function. [Pass]

11.3.8 Testing the setAddress(string address) function
- Checking whether the address of the manufacturer is set to the string address passed to the function. [Pass]

11.3.9 Testing the setPhone_no(Int phoneno) function
- Checking whether the function sets the phone number of the manufacturer to the Int phoneno passed to the function. [Pass]

11.3.10 Testing the save() function

- Insert a new manufacturer to the database if it is not present already [Pass]
- Do not insert the manufacturer to the database if it is present already. [Fail]

11.3.11 Testing the delete() function

- Delete a manufacturer present in the inventory database [Pass]
- Do not delete the manufacturer if it's not already present in the database. [Fail]

11.3.12 Testing the AddItem(Item a) function

- Adding the item to the manufacturer's list of products [Pass]

11.4 Testing the Invoice class

11.4.1 Testing the Constructor Invoice(Date date, float cost_price, float selling_price)
The constructor is called only after ensuring that all the parameters passed are valid. So, they will be tested either in the GUI testing or in the database testing. So, there is no need to test the constructor here.

11.4.2 Testing the addItem(Item a) function

- Adding the item to the item arrayList of the invoice object [Pass]

11.4.3 Testing the getRevenue() function

- Returns the revenue of the order [Pass]

11.5 Testing the Database class

11.5.1 Testing the retrieveItem() function

- Return the item list from the database [Pass]

11.5.2 Testing the retrieveManufacturer() function

- Return the Manufacturer list from the database [Pass]

11.5.3 Testing the retrieveInvoice() function

- Return the Invoice list from the database [Pass]

11.5.4 Testing the addItem(Item a) function

- Adding the item to the item master arraylist [Pass]

11.5.5 Testing the addManufacturer (Manufacturer a) function

- Adding the manufacturer to the manufacturer ArrayList [Pass]

11.5.6 Testing the addInvoice(Invoice invoice) function

- Adding the invoice to the Invoice ArryaList [Pass]

11.5.7 Testing the deleteItem(Item a) function

- Deleting the item from the item ArrayList [Pass]

11.5.8 Testing the deleteManufacturer(Manufacturer a) function

- Deleting the manufacturer from the manufacturer ArrayList [Pass]

12. References

The following documents are referred to support the test plan of our doucment:

1. Software requirement specification document
2. Class Diagram
3. Use case diagram
4. IEEE-829 Test Plan Outline