Test Document

for

Shop Inventory System

Version 1.2

Prepared by

Group #: 20		Group Name:
Kshitij Kabeer	180366	kshitijkabeer@gmail.com
Rishabh Kothary	180608	rishabhkothary76@gmail.com
Kartavya	180343	kartavya4301@gmail.com
Pravar Deep Singh	160508	pravardeepsingh@gmail.com

Course: CS253

Mentor TA: Swastim Maitak

Date: 28th April 2022

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.1	Kshitij Kabeer, Rishabh Kothary	Initial Document	04/04/22
1.2	Kshitij Kabeer Rishabh Kothary	Updates from Beta Testing Phase	28/04/2022

1 Introduction

Mention the test strategy
(Manual testing? Test automation? …)

When was the testing conducted?
(in parallel with the implementation? After the implementation was completed? …)

Who were the testers?
(the developers? Independent testers? …)

What coverage criteria were used?

Have you used any tool for testing?

Our primary test strategy was manual testing which was conducted in parallel with implementation. The testers were the developers and were trying to cover functional coverage. We haven't used any automated testing tools.

2 Unit Testing

1. Forecasting

We were testing if the update forecast method which primarily predicts future demands based on previous demands and previous forecasts was working appropriately. Dummy test data was generated with artificial demand to test the function. Testing was done during implementation

Unit Details: *Class : AdminOptionSelection Function: update_forecast()*

Test Owner: *Rishabh Kothary*

Test Date: *23rd March 2022*

Test Results: *Many test cases were tried to handle every possible edge case. As the function was not very huge we are able to ensure functional coverage.*

Structural Coverage: *Branch coverage, Statement coverage, Condition Coverage, Decision Coverage.*

2. Verifying Credentials

verifyCredentials is a very crucial database function needed for logging in. We added artificial employees and admins to test the function. Testing was done during implementation.

Unit Details: *Class : Database Function: Verify_Credential*

Test Owner: *Kshitij Kabeer*

Test Date: *3rd April 2022*

Test Results: *As the function length was small we were able to handle many edge cases and were able to ensure functional coverage. The possible cases tested are: Both username and password correct (should take the user to the correct page depending on whether he is the admin or a regular employee), only username wrong, only password wrong, both username and password wrong (should not be taken anywhere in the case of the latter three inputs)*

Structural Coverage: *Branch coverage, Statement coverage, Condition Coverage, Decision Coverage.*

3. Optimization Function

The optimization algorithm was built using sound sources. To test proper functional coverage we generated artificial data to see if the algorithm gave sound results.

Unit Details: *Class : RestockingSuggestion Function: optimizer*

Test Owner: *Rishabh Kothary*

Test Date: *20th March 2022*

Test Results: *The algorithm required as input expected forecast, previous demand, buying price, selling price and holding cost. We generated artificial data to mimic real world*

demands and prices to check if the function produced sound results. As the algorithm was inspired by well studied algorithms we did not test the algorithm mathematically rigorously.

Structural Coverage: *Branch coverage, Condition coverage, Statement coverage, Decision Coverage.*

3 Integration Testing

1. Forecasting and Updation of Database

I was testing if the Database functions and forecasting functions had integrated appropriately. Tested after implementation

Module Details: *Integrating the database module with the forecasting modules. Included the AdminOptionSelection class and the Database class*
Test Owner: *Rishabh Kothary and Kshitij Kabeer*
Test Date: *3rd April 2022*
Test Results: *We used SQlite DB viewer to see real- time results of using the forecast button in the software. We tested if the Predict_table table in the SQL database was updating correctly or not. We even kept updating the demand and applied forecasting multiple times and found the module to be working perfectly fine. We also made sure that the Sold_Items table and Predict_table were updated if there were additions/deletions of new items, so that the forecasting functions never had to deal with incomplete data. I also added input sanitisation which I tested by trying to enter erroneous inputs in all fields.*

2. Update Item Information

Integrated Front-end and the Database and I could add or update item information like buying price, selling price, holding cost, changes in stock ,etc. I tested if item information was updated properly in the database, I also let stocks be negative to indicate stocks being bought for future delivery. I also integrated the item-invoice generation to update changes in stocks. Tested after implementations.

Module Details: *Integrated various functions of Database class and the EmployeePortal class.*
Test Owner: *Kshitij Kabeer*
Test Date: *2nd April*
Test Results: *I added various dummy items and also generated many dummy invoices to ensure that item information was updated correctly. However, a few known edge cases are not covered. These cases arise from erroneous inputs on the part of the user, for example, he/she may leave any field blank (in which case an empty string will be stored in the database), or give a string for a real number field like buying price. I also added the feature of deleting the item, which was incomplete, and tested it by trying to delete various items. I also added input sanitisation which I tested by trying to enter erroneous inputs in all fields.*

3. Add/Change Credentials of Employees

Integrated various functions of Database class and ChangeLoginInfo class. I added various dummy employees and dummy admins to ensure proper functional coverage.

Module Details: *Integrated various functions of Database class and the AdminOptionSelection.*

Test Owner: *Kshitij Kabeer*

Test Date: *2nd April*

Test Results: *I added various dummy employees and admins and tested if these employees were able to login with correct access privileges. I also modified each of these dummy employees, and tested whether they are able to login with the new credentials or not, and simultaneously, whether the previous credentials stop working or not. Also did the same test after deleting employees. I also added input sanitisation which I tested by trying to enter erroneous inputs in all fields.*

4 System Testing

1. **Requirement:** *Restocking Function*
   We have tested the individual components of the module, and they are working fine. Component has been integrated properly

Test Owner: *Rishabh Kothary*
Test Date: *4th April 2022*
Test Results: *We generated artificial datasets to test the functionality. The algorithm was created from sound sources and by testing with artificial data, the algorithm seemed to suggest reasonable restocking solutions.*

2. **Requirement:** *Generating Invoice*

This requirement is for the employee, as he has to generate an invoice for each customer.
Test Owner: *Kshitij Kabeer*
Test Date: *2nd April*
Test Results: *I tested the employee interface which was used to generate the invoice. Each and every button was tested for its effect on the invoice. I also made sure that the data being updated in the table was correct, after generating the invoice. In the beta testing phase I fixed the edge case where an item can go out of stock.(Issue #9), and tested it by trying to add a quantity of items without enough stock.*

3. **Requirement:** *Stock Information View*
   This requirement is for the admin, as he has to view the current status of his items.

Test Owner: *Kshitij Kabeer*
Test Date: *1st April*
Test Results: *This was easy to test, just verify whether all the information displayed in the stock information table is the same as in the database*

….

5 Conclusion

How Effective and exhaustive was the testing?
We could have done better testing had we had the time and manpower. Many edge cases have not been tested and accounted for, especially pertaining to erroneous user input. But, we have included most (if not all) edge cases in the manual so that the user can avoid these mistakes.

Which components have not been tested adequately?
The algorithm for restocking suggestions has not been mathematically tested rigorously for correctness and accuracy. Also, as mentioned earlier, sanitisation of user input is not done. We didn't have the time to think of, test, and then code for every edge case that might arise during manual input. Update: I have added some rudimentary level of input sanitisation in the beta testing phase (Details can be seen on github under issue #6).

What difficulties have you faced during testing?
Since the tests are not automated, we had to build the entire application every time, and then run the tests. Also, due to less manpower, we couldn't test each and every edge case.

How could the testing process be improved?
First and Foremost: Adding automated tests. This will ensure that we don't have to build the entire application every time we have to test changes in a function or class.
Secondly: Sanitisation of Input. A lot of errors occur due to erroneous input on the user side. These need to be accounted for and the software needs to be appropriately changed and tested.

Appendix A - Group Log

<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to produce this document>

There has been no meeting, Only Rishabh and Kshitij have tested the software together.