**UNIT TESTING**

1. **Class ShoppingBag**
2. Test Case: check whether the product could be added properly.

Test Data: Execute method - addProduct(new Product(String name, String productCode, String productDescription, String material, String size, double price, int stockAvailability, File picture)) – twice

Expected Result: only one order is created, all items added to the list, items’ information all match, and the total price is correctly shown.

1. Test Case: check whether the product could be deleted properly.

Test Data: Execute method – removeProduct(Product(String name, String productCode, String productDescription, String material, String size, double price, int stockAvailability, File picture)) – twice

Expected Result: the ShoppingBag list is empty, the total price is 0, and the order is removed.

1. Test Case: check whether the order can be submitted properly.

Test Data: addProduct(Product(String name, String productCode, String productDescription, String material, String size, double price, int stockAvailability, File picture));

submitOrder();

Expected Result: order is successfully submitted, and the user is taken to the payment page.

1. **Class Client (Including its Subclasses)**
2. Test Case: check whether a client object could be created properly.

Test Data: Client customer = new Customer(String name, String email, int mobilePhone);

Client employee = new Employee(String name, String email, int mobilePhone, String storeID, String username, String password)

Expected Result: information of two clients are stored correctly in the system.

1. Test case: check whether the boolean ConnectWifi method can identify users’ wifi signal properly.

Test Data: make the customers visit the store main page with their 4G network signal.

Expected Result: boolean wifiConnected field returns true, and all functions can be processed correctly.

1. Test case: send correct and valid customer’s phone build in GPRS system location to findClosestStoreLocation (GPRS gprs) method.

Test Data: input an invalid GPRS string as a parameter of this method.

Expected Result: the closest store locations could not be loaded successfully.

Fail test result: the system might provide some default stores’ descriptions.

1. **Class QRCode**
2. Test Case: check whether the QR code can be created successfully.

Test Data: generateQRCode(getProductCode(new Product (String name, String productCode, String productDescription, String material, String size, double price, int stockAvailability, File picture)))

Expected Result: A new QR code is generated and stored in the system.

1. Test Case: check whether the new QR code can be used properly.

Test Data: matchProduct(QRCode)

Expected Result: the product stored in the system can be found successfully, with all information matching that is stored in the QR code.

1. **Class Message**

(Methods of this class need to be checked with the execution of methods which involves showing messages from other classes)

1. Test Case: check whether messages could be shown.

Test Data: showSuccessMessage();

showErrorMessage();

Expected Result: Correct messages can be shown.

1. **Class Internet\_Detection**
2. Test Case: provide correct user GPRS location via build in GPRS system.

Test Data: call getGPRSlocation() method

Expected Result: a valid location will be sent to IO-Handler.

1. Test Case: ouput right WLAN network IP address.

Test Data: users connect store’s wifi and then running getWifiSignal() method

Expected Result: output an IP address with correct format.

1. **Class IO\_HANDLER**
2. Test Case: void TransformLocation(GPRSLocation) method can transfer the GPRSLocation to a readable information in program.

Test Data: runnning this method and check the final type of data.

Expected Result: the output type of location is String after running TransformLocation() method.

Possible Fail result\_1: Some loctaions are unrecognized. Eg: NULL POINTER ERROR

Possible Fail result\_2: locations can be recognized. But system fail to proceed transformation.

The appearance of crashes.

1. Test Case: transformWifiConnection method can check whether users have connected to a stores’ WLAN Network.

Test Data: running TransformWifiConnection(Signal) method with other stores’ wifi

Expected Result: the IP address could be recognized but the result still output false as well. Because the client connected to the incorrect WLAN network.

Fail result: An error message is popped. The system could not recognize this IP address.

Output null and throws an invalid exception.

Fail result: Output boolean true value as well.

1. **Class STORE**
2. Test Case: when we run getStoreId() , all output Id should be unique.

Test Data: add the store with the same Id repeatively in the list.

Expected result: all duplicated store descriptions and id are deleted automatically.

Fail result: two Store Objects have the same Id.

1. Test Case: check whether the description for each store can be displayed properly.

Test Data: getStoreDescription(getGPRS())

Expected Result: all duplicated store descriptions are removed automatically, and the description for each store matches.

Fail Result: The description of the store does not match.

1. **Class Product**
2. Test Case: A product can be created and store in the system.

Test Data: Product product = new Product(String name, String productCode, String productDescription, String material, String size, double price, int stockAvailability, File picture);

getName();

productCode();

getProductDescription();

getProductMaterial();

getSize();

getPrice();

getStockAvailability();

getProductPicture();

Expected Result: Corresponding values are returned.

Fail Result: No values are returned or values returned do not match the product.

1. **Class Order**
2. Test Case: Check whether a product can be selected and added by the order

Test Data: Product product = new Product(String name, String productCode, String productDescription, String material, String size, double price, int stockAvailability, File picture);

selectProduct(product);

addProduct(product);

Expected Result: The chosen product is added to the ShoppingBag by the Order successfully, with all fields returning the correct values (ie other than orderNum, quantity, price, createDate, the number of shoppingBag which records the specific shoppingBag all products are added to should only be one). At the same time, a success message is shown.

Fail Result: The product is not added due to some errors, and all fields should be null or 0. An error message is shown simultaneously.

1. **Class Account**
2. Test Case: check whether an account is linked to the correct banking details.

Test Data: Account account = new Account(boolean tempOrNot, String email, String address, File bankingInfo);

getBankingInfo();

Expected Result: the correct banking details are returned.

Fail Result: No banking details or wrong banking details are returned.

1. Test Case: check whether an order along with the ShoppingBag is linked to an account properly.

Test Data: getOrder();

getShoppingBag();

Expected Result: The correct Orders and the matching ShoppingBags are returned, with the corresponding products’ information.

Fail Result: No or the wrong Orders and ShoppingBags are returned. Or, if the account has not made any orders, Order(s) and ShoppingBag(s) are returned.

1. **Class Payment**
2. Test Case: only authorized payment can be made.

Test Data: getWhetherAuthorized();

Expected Result: the whetherAuthorized field returns true, and the money is received by the system.

Fail Result: Money is received by the system even when whetherAuthorized returns false. Or the incorrect amount of money is received.

1. Test Case: check whether the payment information is stored correctly.

Test Data: getPaymentDetails();

Expected Result: field Id, paymentDate, and description all return the correct values.

Fail Result: No or wrong values are returned.

**INTEGRATION TESTING**

**Catalog:**

* **Register and Login Process**
* **Inventory manage page**
* **Product view**
* **Place items /remove items in the virtual shopping bag.**
* **Check Out**
* **Creating order.**

1. **Register and Login Procedure**

ACTOR：Clients

According to the requirement analysis, the login interface will be activated when check out.

Test Case：there are some differences between customer’s login panel and employee’s login panel.

According to the use-case description: 

Test Data: use Employee’s Username and Password to login customer Panel.

Pre-condition: WIFI connection is stable in this store. And user’s IP address has been detected successfully.

Expected result: the system should send an “fail message” to the client.

Failed test result: It is surprising to find that employee’s account can login customer’s panel as well.

1. **Inventory Manage page**

ACTOR：Employee, System: Inventory system API

Test Case: Employees are able to view exact availability of items and product information at the inventory manage page.

Test Data: employee update one product’s stock.

Precondition: employee has used their account to log in.

Expected result: after executing this operation, product’s availability is updated to the customer’s

user interface at the same time.

Fail result: the customer panel doesn’t have any response.

Test Case: staff can only load a single inventory manage page with one store Id.

Test Data: make two employees who work in two different stores to enter in the inventory manage page.

Pre-condition: both of their store ids are valid.

Expected result: load two totally different descriptions.

Failed Result: one of them login in an unmatched store description.

1. Main Product page

ACTOR: Customer, System: Inventory system API

Test Case: customer can view exact product information in current store.

Test Data: running key operation getStoreDescription()

Pre-condition: customer has passed network-detection.

Expected Result: product information is consistent with actual info.

1. **Placing items / Removing items in the shopping bag**

ACTOR：Customer, System: InventorySystem API

Test Case: customers can add same products repeatedly.

Test Data: running addProduct() method with the same product and clicking “viewing shopping bag” button.

Expected System Response: to display the product information once, but its quantity increases.

Test Case: customers can remove products from the shopping cart, if they do not like them.

Test Data: execute key operation removeProduct().

Expected Test Result: the items are no longer to display in the virtual shopping bag.

Test Case: the alert message will be sent to customer when the product is not available right now.

And customer has authority to determine whether they want to find items from other shops.

Pre-condition: Assume that customer has logged in successfully.

Test Data: sending alert and if answer is “yes” then executing operation findClosestStores()

Expected Result: As answer is yes, provide search result with customers. As answer is no, Jump to main page.

1. **Check-Out**

ACTOR：Customers, System: Payment system API.

Test Case: Payment service will ask customers to login and submit payment account details.

Test Data: jump to login panel. Executing the operation loadingLogin Page.

Pre-condition: the customer’s virtual shopping cart should not be empty.

Expected Result: login interface is loaded automatically .

Test Case: customer will receive confirmation message, after paying successfully.

Pre-condition: the customer has logged in his account.

Test Data: running program sendConfirmationMessage();

Expected Result: customer’s mobile phone will receive payment confirmation message within 30s.

1. **Create Order**

ACTOR: Customer, System: orderSystem API

Test case: after completing payment process, the order system will synchronize order information on the order Panel.

Test Data: the process system creates a new order.

Expected Result: the customers can view their order specifically.

**SYSTEM TESTING**

**Depends on use-case diagram**

The interaction between payment system service

Test Case: after completing the procedure of payment, the order system will submit order depends on payment confirmation.

Test Data: executing “processPayment” operation and then submit an order.

Expected Result: the order information is consistent with customer’s payment details.

**The interaction between inventory system and order System.**

Test Case: after creating an order, the stock of this product will be updated.

Test Data: running program “view product info”, after creating a valid order.

Expected Result: the stock availability of this item decreases.

**The interaction between inventory system and virtual shopping bag**

Test Case: The number of available products determines whether the item can be placed in virtual shopping cart.

Test Data: attempt to add an unavailable product to the shopping cart.

Pre-condition: customers can acquire inventory information.

Expected Result: an alert message is sent to customer. Customers have authority to purchase this product from the closest shopping stores.