Test Plan of Online Sales Portal

Authors : Group 8
(Anand Manojkumar Parikh, Soni Aditya Bharatbhai, Shashwat Naidu)

1. Test Plan Identifier : Online_Sales_Portal_Test_Plan_1

2. References :

a. Software Requirements Specification
b. Use case diagram, Class diagram
c. IEEE - 829 Test Plan template

3. Introduction :

- This is the unit test plan (Version 0.0.1) for the software Online Sales Portal
- This document aims to test the different methods and functions used in the OSP software.
- Unit tests are provided in this project, for understanding by both - the users and the developers.

4. Test Items (Functions):

(i) Class Address:

a) constructor

—--

(ii) Class Category:

a) constructor
b) getData()

(iii) Class Order:

a) constructor
b) negotiation()
c) update_delivery_status()
d) getData()

(iv) Class Item:

a) constructor
b) getData()
c) search()

(v) Class User:

a) constructor
b) getData()
c) changePassword()
d) changeData()

(vi) Class Seller:

a) constructor
b) add_product()
c) delete_product()
d) view_pending_orders()
e) view_sales()
f) update_order_status()
g) negotiate()
h) getData()

(vii) Class Buyer:

a) constructor
b) raise_purchase_request()
c) negotiate()
d) payment()
e) update_delivery_status()
f) getData()

(viii) Class Manager:

a) constructor
b) change_category()
c) add_category()
d) remove_category()
e) reject_item()
f) manage_seller()
g) manage_buyer()
h) audit()
i) help_negotitations()
j) getData()

—---

5. Software risk issues :

One of the most significant risks is not being able to meet the aggressive deadline allotted to this project. Trying to meet the deadline might lead to a few minor reductions in the scope and features provided, but the quality will not be compromised.
Another issue being faced is inadequate team members, since the Developers are required to do all the things like : Coding (the main task) , managing, testing, etc.

6. Features to be tested :

6.1 User Class features

a) Sign up interface (special key for Manager sign-up)
b) Sign in interface
c) Search interface

6.2 Manager Class features

a) Add/remove categories
b) Change category of a product
c) Reject items of poor quality
d) Manage buyers and sellers
e) Perform audit of purchases

6.3 Buyer Class features

a) Buy interface
b) Negotiation interface
c) Payment interface

6.4 Seller class features

a) Add/remove product items
b) Change product details
c) Negotiation interface

7. Features not to be tested :

- The graphical user interface made using front-end development tools such as Bootstrap, HTML, CSS and Javascript are NOT to be tested.
- The database implemented through MongoDB will NOT be tested.
- The backend written using Flask will NOT be tested.
- Enumeration classes (eg. list of states and cities, calendar) built into the system are not tested (no need to test these as it is very obvious).

—---

The reason NOT to test the above mentioned items is that these are well-known and bug-free libraries and tools. Also, these are not implemented by the developers of this project (Online Sales Portal).

8. Approach :

8.1 Testing levels :

- Unit testing and system/integration testing will be done by the developers.
- Acceptance testing will be done by developers along with TAs and Professors guiding us through this project.

8.2 Test tools :

- For low-level testing (of the source code), a python compiler and an internet connection is required.
- For testing the website once it is hosted, the only requirement is a strong internet connection, and can be done by developers as well as non-developers.

8.3 Meetings :

- The developer team conducts meetings every day in order to keep each other up-to-date and speedily try to achieve all the goals and deadlines provided in this project.

The detailed methods for how exactly the test plans are designed, along with appropriate exception flow are explained below.

TEST PLANS AND SCENARIOS

Class Address :

1. Residence number : String (must be non-empty)
2. Street : String (must be non-empty)
3. Locality : String (must be non-empty)
4. City : Enumeration (Must be chosen from the dropdown box)
5. State : Enumeration (Must be chosen from the dropdown box, City list is provided accordingly)
6. Pincode : Unsigned Integer (Must be a valid 6 digit pincode)

Whenever a constructor is called, the program verifies that all the initialization parameters satisfy the above conditions. If one or more conditions are not satisfied, the constructor must raise an error and print the corresponding details of the error.

Test plan:

1. Call the constructor with valid initialization parameters and print the object’s attributes.

—---

2. Call the constructor with one or more invalid initialization parameters.

Class Category :

1. ID: Integer(system-generated, unique)
2. Name: String(must be non-empty)
3. categoryList: List of Category Objects(static)

Whenever a constructor is called, the program verifies that all the initialization parameters satisfy the above conditions. If one or more conditions are not satisfied, the constructor must raise an error and print the corresponding details of the error.

Test plan:

1. Call the constructor with a valid initialization parameter and print the object's attributes.
2. Call the constructor with an invalid initialization parameter.

Class Item :

1. ID : Integer (System generated, unique)
2. Name : String (non-empty)
3. Seller_id : Integer (will be a valid seller_id from list of sellers)
4. Category : Category object
5. Photo : Image (non-empty)
6. Price : Integer (non-negative)
7. Age : Integer (non-negative)
8. Description : String (optional)
9. Manufacturer_name : String (non-empty)
10. City : Enumeration (chosen from dropdown)
11. Is_heavy : Boolean (default : FALSE)
12. Category_items : Unordered_map / Dictionary (item to category)

The constructor must always check that the initialization parameters satisfy the above conditions. It must raise an error and print the corresponding message in case of invalid initialization parameters.
Search returns "No matching items found" if the product searched (or its category) does not exist in the database.

Test Plan :

1. Call the constructor with valid initialization parameters and print the object's attributes.
2. Call the search item for an item in the database and for an item not in the database.

—---

Class Order :

1. Offer_price : Integer (non negative)
2. Item : Object of class Item (non-empty, must be present in category->item map)
3. Buyer : Object of class buyer (non-empty, must be present in database)
4. Seller : Object of class seller (non-empty, must be present in database)
5. Request_status : Enumeration (pending, approved or rejected, default : pending)
6. Delivery_status : Boolean (default : FALSE)
7. Payment_status : Boolean (default : FALSE)

The constructor must always check that the initialization parameters satisfy the above conditions. It must raise an error and print the corresponding message in case of invalid initialization parameters.
Negotiation method is called from the negotiation methods of class Buyer and Seller. If either of them provide a negative offer price, an exception is raised and negotiation price must NOT be updated.

Test Plan :

1. Call the constructor with valid initialization parameters and print the object's attributes.
2. Call Negotiation method, once from both classes buyer and seller, with valid and invalid inputs. For valid inputs, it must update the offer price of the order.
3. Once the buyer calls update_delivery_status, the delivery status of this order must also get updated.

Class User :

1. UserID : Integer (System generated, unique)
2. Password : String (Non-empty)
3. Name : String (Non-empty)
4. Email : String (must be valid email address)
5. Telephone : Unsigned Integer (10 digit valid phone number)
6. Address : Object of address class (handled by Address class)

The constructor must always check that the initialization parameters satisfy the above conditions. It must raise an error and print the corresponding message in case of invalid initialization parameters.
The changePassword function must first ask the user to enter the old password and verify it before allowing the user to update the password.
The changeData function allows the user to change name, email, telephone and address, but not the user-id.

Test plan:

1. Call the constructor with valid initialization parameters and print the object's attributes.

—--

2. Call the constructor with one or more invalid initialization parameters.
3. Call the changePassword function with the old password and new password(should raise errors when required).
4. Change the data by inputting new valid parameters and check if it is updated

Class Manager :

1. Inherited from class User, so parameters of parent class checked by Class User
2. Gender : Enumeration (chosen from dropdown)
3. Date of birth : Select from provided calendar
4. Sign-up key : The correct sign-up key must be provided before making a new Manager account

The constructor must always check that the initialization parameters satisfy the above conditions. It must raise an error and print the corresponding message in case of invalid initialization parameters.
All exceptions while adding, removing and changing categories are handled by the category class.

Test plan :

1. Call the constructor with valid initialization parameters and print the object's attributes.
2. Call the constructor with invalid initialization parameters.
3. Call the change category function on an existing item and print the item's category to check the update.
4. Call the add_category function and print the list of categories to check the update.
5. Call the remove_category function for an existing category and then print the list of categories to check the update.
6. Call the reject_item function for an existing product item and then remove the item. Verify update by searching for the item.
7. Call the manage_seller for an existing seller and then remove the seller. Verify the update by searching for the deleted seller in the list of sellers.
8. Call the manage_buyer function for an existing buyer and then remove the buyer. Verify the update by searching for the deleted buyer in the list of sellers.
9. Call the audit function, it must print details of all the transactions performed so far.
10. Call the help_negotitations function for a pending purchase request, it must print the contact details of the corresponding buyer and seller involved in the negotiation.

Class Buyer :

1. Inherited from class User, so super parameters checked by Class Manager

—---

2. Purchase requests : Choose an item and send a buy request to the Seller with a valid offer
3. Purchase history : Handled by system

The constructor must always check that the initialization parameters satisfy the above conditions. It must raise an error and print the corresponding message in case of invalid initialization parameters.
Raise purchase request throws an exception when the request price is less than 0.
Negotiate also throws an exception when the request price is less than 0.

Test Plan :
1. Call the constructor with valid initialization parameters and print the object's attributes.
2. Raise a purchase request with an offered price, once with positive and once with negative amount, and check if it is added to list of pending orders
3. Negotiate with the seller by offering - a positive price (negotiation price successfully replaced) and a negative price (exception raised)

Class Seller :
1. Inherited from class User, so super parameters checked by Class Manager
2. Items : Add / remove / update this list of items on sale
3. Purchase requests : Handled automatically when buyers send requests
4. Sales history : View previous sales made

The constructor must always check that the initialization parameters satisfy the above conditions. It must raise an error and print the corresponding message in case of invalid initialization parameters.
Add_items and Update_items throw an exception (handled by class items) when an invalid entry is entered in any field.
Negotiate throws an exception when the request price is less than 0.

Test Plan
1. Call the constructor with valid initialization parameters and print the object's attributes.
2. Call add_product and add an item once with valid and once with invalid entries. Then check the list of uploaded items by calling viewing uploaded items.
3. Call delete_product and delete a previously uploaded product, and check if it is removed from the list of uploaded items.
4. Call view_pending_orders to find if any buyer has requested a purchase.
5. Negotiate with the seller by offering - a positive price (negotiation price successfully replaced) and a negative price (exception raised).
6. Accept an offer and successfully complete the transaction. Then, check the balance to see if the same amount has been cashed in. Call view_sales function

—---

to see if a sale has been added to the list. Also re-check the list of items and see if that product has been removed.

9. Item pass/fail criteria :

- If the outputs match the provided golden outputs, then it is considered as “Passed”, otherwise “Failed”.
- Proper exception classes will be provided to throw and resolve different kinds of exceptions.
- Efficacy will be based on % of tests passed.

10. Suspension criteria and resumption requirements :

The testing of the software will be suspended based on the following criteria :

1. The actual outputs do not match the golden (expected) outputs.
2. A fatal and unexpected error occurs, such as crashing of the server or database.
3. The website stops responding to user interaction altogether.

The testing of the software is resumed ONLY after all the above issues have been successfully resolved and do not occur again.

No amount of bugs/ defects is acceptable in the delivery of the final software.

11. Test deliverables :

- Test plan
- Test suite

12. Remaining test tasks :

TASK	Assigned To	Status
Create Test Plan	Dev Team	Finished
Define Unit Test Cases	Dev Team	Finished
Verify prototypes of Screens	Dev Team	Pending
Verify prototypes of Reports	Dev Team	Finished

13. Environmental Needs :

The following elements are required for overall testing effort for the Online Sales Portal:

—---

- For low level testing access to python 3 compiler is required, for making the codes and repeated testing of it.
- For high level testing access to an up to date web browser and robust internet connection is required.
- Test data (test cases) have been designed manually by the developers, such that all features and sections of the software are exhaustively tested, to find out and eliminate any bugs whatsoever.

14. Staffing and training needs :

14.1 Training given to users

- The Managers need some prior training on the portal to navigate it easily. However, no special sources are required. All explanations will be provided in the help section available in the portal itself (after logging in).
- The Sellers and Buyers do not need any prior training to use the website. Reading the help section and policies carefully will be sufficient to effectively use the portal.

14.2 Training given to developers

Any developer who wishes to do detail testing of the software, will require to have experience in the following fields :

- Python 3 language
- PyMongo library (for database)
- Flask library (for backend)
- CSS, Javascript, HTML, Bootstrap (for frontend)

15. Responsibilities :

	Dev Team	**Client**
System Integration test execution	X	
Unit test Documentation and execution	X	
System Design Reviews	X	X
Detail Design Reviews	X	
Test Procedure and rules	X	
Screen & Control Prototype Reviews	X	X

—---

Change control and regression testing	X	X

16. Schedule

Time has been allocated within the project plan for the following testing activities.

A. Review of Requirements document by the complete team (and then to be validated by the Professor and the TA) and initial creation of Inventory classes, sub-classes and objectives.
B. Development of Master test plan with time allocated for at least two reviews of the plan.
C. Integrating various modules involved and running System Integration tests.
D. Running various design related reviews to ensure that the design proposed will meet the specified requirements.
F. Unit test time within the development process.
G. Time allocated to correct any small discrepancies, with the help of feedback provided by the Professors and TAs.

17. Planning risks and contingencies

We have provided some contingency plans with regards to a few issues which might occur. They are as follows:

A. The payment done by the buyer does not directly go to the seller. It first stays with the management of OSP. When the buyer approves that he/she has received the product, money is transferred to the seller. This helps avoid any counterfeit seller and makes the portal more secure and reliable to use.

B. Random people cannot sign up as managers as they need to have a unique passcode to sign-up. This stops people from making fake manager accounts and disrupting the online portal.

18. Approvals

Development Manager 1	Anand Manojkumar Parikh
Development Manager 2	Soni Aditya Bharatbhai
Development Manager 3	Shashwat Naidu

—---