Guest-Visitor Scenarios

* Use-Case: Login

1. **Actor:** guest-user
2. **Precondition:**

The user is registered to the system.

1. **Parameters:** username, password
2. **Actions:**
3. System: system verifies that there is a match between the given password, to the user's password stored in the system.
4. If there is a match, the login succeeds, and the guest-user is now logged-in and changes status to **subscribed-user**.
5. If the given password is wrong the use-case failed, and the system presents appropriate error message.

* Use-Case: Register

1. **Actor:** guest-user
2. **Precondition:**

None empty password.

1. **Parameters:** username, password (+ additional personal info? Or only at checkout?)
2. **Actions:**
3. System: validates given username doesn’t already exist in the system.
4. If the username doesn't already exist, the register succeeds, and the guest-user registered to be **subscribed-user**.
5. If the username exists, the system presents appropriate error message to the user and the user is requested to choose different username. Means, the user must repeat the use-case.

Subscribed-Visitor Scenarios

* Use-case: Logout

1. **Actor:** subscribed user
2. **Precondition:**

The user is registered and logged-in to the system.

1. Parameters: None.
2. **Actions:**
3. System: The user is logged-out of the system and the system changes its status back to **guest-user**. User's shopping cart is saved.

* Use-Case: Open Shop

1. Actor: subscribed user
2. Precondition:

The user is registered and logged-in to the system.

The given shop name doesn't already exist in the system.

1. Parameters: shopName
2. Actions:
3. System:
4. If the shopName doesn't exist in the system, creates new shop with shopName. The system presents an appropriate success message to the user and appoints the user as **shop-founder**. Also, the user is the first **shop-owner.**
5. If the shopName exists, the system presents appropriate error message to the user and the user is requested to choose different shopName.

Trade-System Manager Scenarios

* Use-Case: Shop purchase's history report

1. **Actor:** system manager
2. **Precondition:**

The user performing the action is logged in and has system manager permissions.

The requested shop exists in the trade system.

1. **Parameters:** shopName
2. **Actions:**
3. System: presents all the history of the shop purchases.
4. User: approves

* Use-Case: User purchase's history report

1. **Actor:** system manager
2. **Precondition:**

The user performing the action is logged in and has system manager permissions.

The requested user (for the report) registered in the trade system.

1. **Parameters:** username
2. **Actions:**
3. System: presents all the history of the user purchases.
4. User: approves

|  |  |  |
| --- | --- | --- |
| Use-Case | Parameter | Expected Output |
| Register | Username that doesn’t exist in the system. | Success |
|  | Username that already exist in the system. | Fail |
| Login | Valid username (registered in system), password matches to the password in the system. | Success |
|  | Username that doesn’t exist in the system. | Fail |
|  | Valid username (registered in the system), password mismatch. | Fail |
| Logout | The user requesting to logout is logged-in (and registered). | Success |
|  | The user requesting to logout isn't registered to system – username doesn't exist. | Fail |
|  | The user requesting to logout isn't logged-In to the system. | Fail |
| Open Shop | ShopName doesn't already exists in the system, user is registered and logged in. | Success |
|  | User isn’t registered to the system (user doesn’t exist) | Fail |
|  | User is registered but isn’t logged in. | Fail |
|  | User is registered and logged in, ShopName already exist. | Fail |
|  | User tries to add to shop invalid product (one that doesn’t exist in the storage) | Fail |
|  | User chooses invalid purchase policy for certain product. (Policy doesn’t exist) | Fail |
| Shop purchase's history report | The given parameter shopName exists in the system as valid shop, the user performing the action is logged in and has system manager permissions. | Success |
|  | The user requesting the action isn't logged in. | Fail |
|  | The user requesting the action logged in and doesn’t have system manager permissions. | Fail |
|  | The given parameter shopName doesn't exist in the system. | Fail |
| User purchase's history report | The given parameter username exists in the system as registered user, the user performing the action is logged in and has system manager permissions. | Success |
|  | The user requesting the action isn't logged in. | Fail |
|  | The user requesting the action logged in and doesn’t have system manager permissions. | Fail |
|  | The given parameter username doesn't exist in the system. | Fail |

Capabilities and Responsibilities of Classes

* Market – represents the whole system.

This class interacts with 'Users' package - creates the system manager, users are visitors in the market.

Also, the class creates an external connection manager.

* ExternalConnector (singleton) – has a responsibility for creating and managing external connections, with authorized supply systems and payment systems.
* User – main entity in our system. Represent a visitor in the market, and a shopper in the market's shops. We have several kinds of user states, that defines the capabilities and responsibilities available for him.
* Guest – state of user. Has all the capabilities and responsibilities of guest-visitor as mentioned in the general requirement document.
* Subscriber – state of user. Has all the capabilities and responsibilities of subscriber-visitor as mentioned in the general requirement document.
* System-Manager- subclass of subscriber. Special kind of subscriber-user, that has maximum permissions available in the system.
* Shop – main entity in our system. Has responsibilities for managing purchase and discount policy and enforcing them in user's purchases.

In addition, has responsibility in completing purchases process: receive approval for payment and supply of the items, save of the purchases, updating the inventory about the items bought in the purchase.

* ShopPolicy – manages all the policy related to the shop – discounts for products, purchase policy for products and products prices.
* Inventory – manages all the inventory of the system. With every purchase completed, has the responsibility for updating the product's available units in the inventory. Also, the class must not allow completing order of product with X units (for example) when there are less then X units of the product in the inventory.
* Product – represent a general product, each shop can choose whether to sell it or not.