**Use Cases**

**Use-case** Supply

**Actor:** Shop, Market

**Parameters:** items to delivery, client’s address.

**Pre-Condition:** payment was successful.

**Post-Condition:** Supplier will send the delivery.

**Actions:**

1. **Shop** verified that payment was received.
2. **Shop** sends a request to the market to supply a delivery to a client address with his bought item.
3. The **Market** search an available supplier to the delivery and transfer the request to him using the external services.
   1. If there are no available suppliers, the delivery request is failed and a notification to the user will be sent to try again later, the initial process of the order will be kept.
4. The supplier answers that the delivery will be send to the client.
   1. If the supplier cannot deliver the delivery, he will return to the Market that the delivery was not created, and the Market will choose another supplier.
   2. If all the suppliers cannot deliver the requested items, the Market will return to the Shop that the request to create a delivery cannot be done, and a notification will be sent to the user to try again later, and the initial process of order will be kept.
5. The **Market** update the **Shop** that the delivery was created.
6. The **Shop** will send a notification to the **User.**

|  |  |  |
| --- | --- | --- |
| Use Case | Parameter | Expected output |
| Supply | Market with 0 supplier  A delivery of at-least 1 item and an address | Fail |
| Market with 1 available supplier  A delivery of at-least 1 item and an address | Success |
| Market with 2 unavailable suppliers  A delivery of at-least 1 item and an address | Fail |

**Use-case** Add Item to Inventory

**Actor:** Shop Owner, Shop, Inventory

**Parameters:** Item, Quantity.

**Pre-Condition:** Shop owner logged in to his account.

**Post-Condition:** change the inventory.

**Actions:**

1. The **Shop Owner** request the **Inventory** of his **Shop**.
2. He sends a request to the **Inventory** to add the Item with the given Quantity.
   1. If the Item is already existing in the inventory, the Inventory will change the quantity of the Item.
   2. If the quantity is not a positive integer, then the Inventory will alert the user and don’t add the Item.
3. The **Item** is added to the **Inventory** with the given Quantity.

|  |  |  |
| --- | --- | --- |
| Use Case | Parameter | Expected output |
| Add Item to Inventory | Shop with Inventory {Item: 1, Quantity: 3}  Owner request to add item 2 with quantity 1 | Success |
|  | Shop with Inventory {Item: 1, Quantity: 3}  Owner request to add item 2 with quantity 0 | Failure |
|  | Shop with Inventory {Item: 1, Quantity: 3}  Owner request to add item 2 with quantity -1 | Failure |
|  | Shop with Inventory {Item: 1, Quantity: 3}  Owner request to add item 1 with quantity 1 | Success |
|  | Shop with Inventory {Item: 1, Quantity: 3}  Owner request to add item 1 with quantity 0 | Failure |
|  | Shop with Inventory {Item: 1, Quantity: 3}  Owner request to add item 1 with quantity -1 | Failure |

**Use-case** Remove Item from Inventory

**Actor:** Shop Owner, Shop, Inventory

**Parameters:** Item.

**Pre-Condition:** Shop owner logged in to his account.

**Post-Condition:** change the inventory.

**Actions:**

1. The **Shop Owner** request the **Inventory** of his **Shop**.
2. He sends a request to the **Inventory** to remove the **Item**.
   1. If the **Item** does not exist in the **inventory** the inventory will alert the user.
3. The **Item** is removed from the **Inventory**.

|  |  |  |
| --- | --- | --- |
| Use Case | Parameter | Expected output |
| Remove Item from Inventory | Shop with Inventory {Item: 1, Quantity: 3}  Owner request to remove item 1 | Success |
| Shop with Inventory {Item: 1, Quantity: 3}  Owner request to remove item 2 | Failure |

**Use-case** Change Item’s Detail

**Actor:** Shop Owner, Shop, Inventory, Item

**Parameters:** Item.

**Pre-Condition:** Shop owner logged in to his account.

**Post-Condition:** Item is Modify.

**Actions:**

1. The **Shop Owner** request the **Inventory** of his **Shop**.
2. He sends a request to the **Inventory** to change the detail of **Item**.
   1. If the Item does not exist in the inventory, the process will stop, and the Inventory will notify the user.
3. The **Owner** chooses the detail to change.
   1. If the detail that the **Owner** has chosen is not legal the system will alert him and don’t change them.

|  |  |  |
| --- | --- | --- |
| Use Case | Parameter | Expected output |
| Change Item Details | Shop with Inventory {Item: {ID:1, category: dairy}, Quantity: 3}  Owner request to change the category of item 1 to bread | Success |
| Shop with Inventory {Item: 1, Quantity: 3}  Owner request to change the category of item 2 | Failure |

1. The **Item** modify the given details.

**Use-case** Change Buying Shop Policy

**Actor:** Shop Owner, Shop, Shop Buying Policy

**Parameters:** none.

**Pre-Condition:** Shop Owner logged in to his account.

**Post-Condition:** Shop Policy is modified.

**Main-Scenario:**

1. The **Shop Owner** request to change the shop buying policy from the **Shop.**
2. The **Shop Owner** view the current buying policy and decide which aspect.
3. The **Shop Policy** is modifying.

**Alternative-Scenario:**

1. If the new policy contradicts the consistency policy given to the founder the shop policy won’t be modified and an alert will be given to the user.

**Use-case** Change Discount Shop Policy

**Actor:** Shop Owner, Shop, Shop Policy

**Parameters:** new discount policy value.

**Pre-Condition:** Shop Owner logged in to his account.

**Post-Condition:** Shop Policy is modified.

**Main-Scenario:**

1. The **Shop Owner** request to change the shop policy from the **Shop.**
2. The **Shop Owner** request to change the discount policy of the shop.
3. The **Shop Owner** send the new value of the policy.
4. The **Shop Policy** is modifying.

**Alternative-Scenario:**

1. If the new policy contradicts the consistency policy given to the founder the shop policy won’t be modified and an alert will be given to the user.

**Use-case** Change Item’s buying Shop Policy

**Actor:** Shop Owner, Shop, Shop Policy

**Parameters:** items id, new buying policy value.

**Pre-Condition:** Shop Owner logged in to his account.

**Post-Condition:** Shop Policy is modified.

**Main-Scenario:**

1. The **Shop Owner** request to change the shop policy from the **Shop.**
2. The **Shop Owner** request to change the buying policy of the items in the shop.
3. The **Shop Owner** send the new value of the policy and the items id.
4. The **Shop Policy** is modifying.

**Alternative-Scenario:**

1. If the new policy contradicts the consistency policy given to the founder the shop policy won’t be modified and an alert will be given to the user.
2. If the item doesn’t exist in the shop Inventory the Owner will be alerted.

**Use-case** Change Item’s Discount Shop Policy

**Actor:** Shop Owner, Shop, Shop Policy

**Parameters:** items id, new discount policy value.

**Pre-Condition:** Shop Owner logged in to his account.

**Post-Condition:** Shop Policy is modified.

**Main-Scenario:**

1. The **Shop Owner** request to change the shop policy from the **Shop.**
2. The **Shop Owner** request to change the buying policy of the items in the shop.
3. The **Shop Owner** send the new value of the policy and the items id.
4. The **Shop Policy** is modifying.

**Alternative-Scenario:**

1. If the new policy contradicts the consistency policy given to the founder the shop policy won’t be modified and an alert will be given to the user.
2. If the item doesn’t exist in the shop Inventory the Owner will be alerted.

**Use-case** Appoint New Shop Owner

**Actor:** Shop Owner, Shop

**Parameters:** User ID to appoint.

**Pre-Condition:** Shop Owner logged in to his account.

**Post-Condition:** the shop has new shop owner.

**Actions:**

1. The **Shop Owner** enter to his shop setting and choose to add new shop owner.
2. **The Shop Owner** enter the **User** ID of the user he wants to appoint.
3. The **System** will check that the User ID belong to a registered User.
   1. If the User ID doesn’t belong to a registered User, the system will return a notification to the owner.
   2. If the User ID belong to an owner of another shop, the system will return a notification to the Shop Owner and the User ID won’t be added to the Owners list.
   3. If the User ID belong to an owner of another shop, the system will return a notification to the Shop Owner and the User ID won’t be added to the Owners list.
4. The User ID will be added to the Owners list of the shop.
5. The user will be notifying about that.

|  |  |  |
| --- | --- | --- |
| Use Case | Parameter | Expected output |
| Appoint New Shop Owner | Shop with Owners list [1,2]  Owner 1 request to appoint user 3 to be owner | Success |
| Shop with Owners list [1,2]  Owner 1 request to appoint non-user 3 to be owner | Failure |
| Shop with Owners list [1,2]  Owner 1 request to appoint user 2 to be owner | Failure |
| Shop with Owners list [1,2]  Shop2 with Owners list [3]  Owner 1 request to appoint user 3 to be owner. | Failure |

**Use-case** Appoint New Shop Manager

**Actor:** Shop Owner, Shop

**Parameters:** User ID to appoint.

**Pre-Condition:** Shop Owner logged in to his account.

**Post-Condition:** the shop has new shop Manager.

**Actions:**

1. The **Shop Owner** enter to his shop setting and choose to add new shop owner.
2. The **Shop Owner** enter the **User** ID of the user he wants to appoint.
   1. If the User ID doesn’t belong to a registered User, the system will return that to the Shop Owner.
   2. If the User ID belong to an owner or manager of another shop, the system will return that to the Shop Owner and the User ID won’t be added to the mangers list.
   3. If the User ID already belong an Owner or a Manager of this Shop, the system won’t add it again and alert the shop owner of such.
3. The **System** will check that the User ID belong to a registered User.
4. The User ID will be added to the Owners list of the shop.

|  |  |  |
| --- | --- | --- |
| Use Case | Parameter | Expected output |
| Appoint New Shop Manager | Shop with Owners list [1,2] and manager list []  Owner 1 request to appoint user 3 to be manager | Success |
| Shop with Owners list [1,2] and manager list []  Owner 1 request to appoint user 2 to be manager | Failure |
| Shop with Owners list [1,2] and manager list []  Owner 1 request to appoint non-user 3 to be manager | Failure |
| Shop with Owners list [1,2] and manager list [3]  Owner 1 request to appoint user 3 to be manager | Failure |
| Shop with Owners list [1,2] and manager list []  Shop2 with Owners list [3] and manager list []  Owner 1 request to appoint user 3 to be manager | Failure |
| Shop with Owners list [1,2] and manager list []  Shop2 with Owners list [3] and manager list [4]  Owner 1 request to appoint user 4 to be manager | Failure |

**Use-case** Change Manager Privileges

**Actor:** Shop Owner, Shop

**Parameters:** User ID of a manager.

**Pre-Condition:** Shop Owner logged in to his account.

**Post-Condition:** the permission of the manger is changes.

**Actions:**

1. The **Shop Owner** enter to his shop setting and choose to add new shop owner.
2. The **Shop Owner** enter the User ID of the user he wants to appoint.
3. The **Shop** will check that the User ID belong to a manager.
   1. If the User ID doesn’t belong to a manger of the shop, the system will alert the Shop Owner.
4. The **System** will show all the permission of the manager with the User ID.
5. The **Shop Owner** select the permission he wants to remove or add to the manger.
   1. If the permission the shop owner choose is illegal the system will alert the Shop Owner and don’t change the permission.
6. The permission of the manager will change, and he will be notifying about that.

|  |  |  |
| --- | --- | --- |
| Use Case | Parameter | Expected output |
| Change Manager Privileges | Shop with Owners list [1,2] and manager list [3]  Owner 1 request to change manager 3 from getting notification | Success |
| Shop with Owners list [1,2] and manager list []  Owner 1 request to change manager 3 from getting notification | Failure |