**Use Cases – תרחישי שימוש**

System + System Administrator use cases:

**Use case – 1**: **System initialization**

1. **Actors:** System.

2. **Pre-conditions:** None

3. **Post-Conditions:** The system is up and running, and users can log in and use it.

4. **Input parameters:** System manager user information, different services information (how to connect to Visa, to Suppliers etc.)

5. **Actions:**

1. The system asks for user information for the system manager.
2. Authentication – The system checks if the user is subscribed, and if so, sets him as the system manager.
3. The system connects to each service provider and checks if the service is up and ready to accept messages.
4. If Everything is working the system moves to the main screen, and ready to handle different users and stores.

6. **Alternatives/Extensions:**

1. If the user information are not correct, or the user given is not subscribed, the system asks again for details and not moving to the next screen.
2. If one of the services fail to receive messages, the system will send notification to the system manager and will halt on the “waiting for the services screen”.

7. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The system manager try to login with a wrong user information. | Invaild user information,  valid services information | Failed authentication, a matching error message will be returned. | Bad |
| The system manager try to conncect to the wrong services (payment, suppliers). | Valid user information, Invalid services information | Failed services connection, a matching error message will be returned. | Bad |
| The system manager was able to connect to the system and initialize the different services. | Valid user information, valid services information | Success, the system will initialize properly. | Good |

**Use case – 2: Change services**

1. **Actors:** System.

2. **Pre-conditions:** The new service exists and is ready to receive messages.

3. **Post-Conditions:** The system can use the new service as usual.

4. **Input parameters:** Information about new service (id, type of service etc.)

5. **Actions:**

1. The system connects to the new service and checks if it is ready.
2. When the new service is ready the system removes the old service.
3. The system sets the new service as the default service for the correct type (Pay, supply etc.)

6. **Alternatives/Extensions:**

1. If the new service fails to start or the information is incorrect, the system will not let the users to remove the old service.

7. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The system try to use a invalid service | Invaild information about the new service | Failed authentication, a matching error message will be returned. | Bad |
| The system try to change the specific service while the service it handles other transcations. | Valid information | Failed services update, a matching error message will be returned. | Bad |
| The sysem received correct data and will update the system. | Valid information | Success, the service will update properly. | Good |

**Use case – 3:** **Using the services (Pay, Supply)**

1. **Actors:** System, Users.

2. **Pre-conditions:** The information that should be sent to the service was collected or created correctly.

3. **Post-Conditions:** The service accepted the request and did what it had to do successfully.

4. **Input parameters:** The parameters that should be sent to the service. (User information, card info, user address etc.)

5. **Actions:**

1. The system validates that all the parameters are correct.
2. The system calls the service using the correct API and passes all the parameters.
3. If the received answer is positive, show a message to the user that the action was done successfully.

6. **Alternatives/Extensions:**

1. If some of the arguments that were received from the user are incorrect, error message will be shown to the user, and he will be asked to fill this parameter again.
2. If the service return a negative answer, handle it by the reason. (Wrong information – Correct it. Service too busy – Make the request later, etc.)

7. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The service try to process an incorrect transaction. | Invaild transaction details | Failed authentication, a matching error message will be returned. | Bad |
| The service try to charge an invalid card details. | Valid transaction details, Invalid card info | Failed  Charge failed, a matching error message will be returned. | Bad |
| The service was unable to process the payment. | Valid transaction details, valid card info | Failed  Charge failed, a matching error message will be returned. | Bad |
| The payment service was able to charge the card succesfully. | Valid transaction details, Valid card info | Success, the payment was successfully received. | Good |

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The service try to process an incorrect supply transaction. | Invaild transaction details | Failed authentication, a matching error message will be returned. | Bad |
| The service try to supply the wrong packages. | Valid transaction details, Invalid package information | Failed  supply failed, a matching error message will be returned. | Bad |
| The service was unable to send a delivery confirmation receipt. | Valid transaction details, valid package information | Failed  supply failed, a matching error message will be returned. | Bad |
| The supply service was able to process the pacakges info and to send a confirmation receipt. | Valid transaction details, Valid package information | Success, the supply was successfully received. | Good |

**Use case – 4:** **Pop up messages**

1. **Actors:** System.

2. **Pre-conditions:** A subscribed user exists.

3. **Post-Conditions:** The user receives a pop-up message in real time or when he enters the system.

4. **Input parameters:** User phone number, the message

5. **Actions:**

1. Check what is the type of action, and if the system has to send the message to his phone number or save it and show it to the user when he enters the system.
2. According to the result of the check, send notification if needed, and save it in the correct database.
3. When the user enters the system, check if there are any new messages that need to be shown to him.

6. **Alternatives/Extensions:**

1. If the system is unable to send pop-up message, ask from the user for the correct phone number or permissions.

7. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The system try to send a message to the wrong user. | Invaild user information | Failed authentication, a matching error message will be returned. | Bad |
| The system try to send a wrong message | Valid user information,  Invalid message | Failed to pop up massages, a matching error message will be returned. | Bad |
| The system received corrcet data and will send it to the given user. | Valid user information,  Valid message | Success, the system will send the message to the user properly. | Good |

**Use case – 5:** **Market history**

1. **Actors:** User who are system manager.

2. **Pre-conditions:** Market working and there are stores and buyers.

3. **Post-Conditions:** The System manager has all the information about the stores and user’s activities.

4. **Input parameters:** Store information, or the buyer information. (id, name, etc.)

5. **Actions:**

a) The system manager writes as the input the ID of the store or buyer that he wants to check the history of.

b) If the ID is correct, the system accesses the database and assembles a new file which contains all the information about the correct object. (Store, User…)

c) The system returns the file with the relevant data to the store manager.

6. **Alternatives/Extensions:**

a) If the information is incorrect, the system asks for the correct information again.

b) If the system fails to assemble the file, it shows proper error message.

7. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The manager try to see the purchases history of an invalid user. | Invaild information | Failed authentication, a matching error message will be returned. | Bad |
| The system manager requests to see the purchases history and at the same time another purchase is made in the same account, so he will not receive an updated histoty. | Valid information | Failed to receive the updated history, a matching error message will be returned. | Bad |
| The system received corrcet information and the manager is able to see the purchases history. | Valid information | Success, the manager will see the history that he wanted. | Good |

Store Visitor use cases (subscriber, guest):

***Use case - 1: guest add searched product to cart***

1. **Actors:** Guest User
2. **Pre-conditions:**
   * none
3. **Post-conditions:**
   * + Products amount did not change
4. **Input Parameters:**
   * Search string params (key words)
   * Product filters string params (price range, ratings, category, name)
   * Store filters string params (ratings, name, id )
5. **Actions:**

a. Guest enters the system

b. Guest search product / store by using the search filters as he wants.

c. System authenticate the guest's token

d. System displays available items matching the search parameters.

e. Guest adds desired item(s) to the shopping cart.

1. **Scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| Product added to cart | At least one of the given input parameters | Product added to cart | Happy |
| Empty search result | At least one of the given input parameters | Output message "no results" | Alternative |
| Authentication went wrong | token | Output message "authentication failed" | Alternative |
| Product unavailable | At least one of the given input parameters | Output message "order invalid" | Bad |

***Use case - 2: guest purchase process***

1. **Actors:** Guest User
2. **Pre-conditions:**
   1. User already added at least 1 product to cart
   2. Order is valid (each product quantity is in stock)
3. **Post-conditions:**
   1. **Success:**
      1. All purchased items are deducted from store's inventory.
      2. Payment from the customer to the shop's owner is successfully processed.
   2. **Failure:**
      1. System reverts to pre-purchase state (atomic action).
4. **Input Parameters:**
   1. User payment parameters (card number, card date, card cvv)
5. **Actions:**
   1. Guest in the cart checkout window to review items.
   2. Guest provides payment information to complete the purchase.
   3. System validates payment information and processes the order.
      1. Validation failed - return to shopping cart
      2. Validation Succeed - continue
   4. Guest exits the system, with the shopping cart cleared.
6. **Scenerios** :

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| Payment authentication fail | Payment information | Output message "payment info invalid" | Alternative |
| Product unavailable | Payment information | Output message "transaction failed" | Bad |
| The guest bought the products | Valid search params, valid payment information | an order was created for the chosen products,  products reduced in the right store by the right amount,  the transaction went through | Happy |

***Use-Case 3: Subscriber shop opening process***

1. **Actors:** Subscriber (Founder)
2. **Pre-conditions:**
   * Subscriber is logged in to the system.
3. **Post-conditions:**
   * New store created.
   * Subscriber assigned as founder.
4. **Input Parameters:**
   * User token
   * Store name
   * contact information: email, phoneNumber, address
   * store description
   * Inventory
5. **Actions:**
   1. Subscriber initiates the process to open their store.
   2. Subscriber provides store name, description, contact information.
   3. Program creates a new store, and turn Subscriber into a Founder
   4. Subscriber completes the shop opening process and now owns a shop
6. **Scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| Outdated Token | User credentials | an error message pops stating "User's credentials unavailable" | Alternative |
| Shop name is taken | Store name | an error message pops stating "shop name is already taken" | Alternative |
| User opens a store | Store name, contact information, store description | Success, a shop was created for the given user | Happy |

***Use case - 4:*** ***Log in***

1. **Actors:** User
2. **Pre-conditions:**
   1. User is registered to the system
3. **Post-conditions:**
   1. User token is generated for this user
4. **Input Parameters:**
   1. Username
   2. password
5. **Actions :**
   1. User enters the system as a guest
   2. User log in to the system
      * 1. User successfully logged in
        2. User failed to log in

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| User logged in successfully | Username, password | User is now recognized by the system as a subscriber | Happy |
| User typed wrong password / username | Username, password | Output message "order invalid" | Alternative |
| There where connection problems during authentication | none | Output message "connection failed" | Bad |

1. **Scenarios :**

Store Owner use cases:

Use case: **4.1** – **Inventory management** (Add product, Remove product, Update product)

1. **Actor**: Store Owner
2. **Pre-condition**: The store owner exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, the product is added/removed/updated according to the given parameters.
4. **Input parameters**: StoreID, Product information (name, price, ...)
5. **Actions**:
6. Store Owner attempts to add/remove/update a product in a specific shop.
7. Authentication – the system checks that the action was requested by an actual store owner of the needed store.
8. Inventory availability – the system checks that the product is available to the requested action in the inventory. (exists if the action is remove/update, not exists if the action is add)
9. Validity – the system checks that the given parameters are valid (positive,non-empty) and follow the standard rules of the store.
10. The system updates the item in the inventory.
11. The system returns a matching success response to the user.
12. **Alternatives/Extensions**:
13. If the authentication failed, a matching error message will be returned and the action will fail.
14. If the item is not available in the inventory, a matching error message will be returned and the action will fail.
15. If the given parameters are invalid (for example zero or empty) or do not follow the standard rules of the store, a matching error message will be returned and the action will fail.
16. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The requesting store owner does not have the right permissions (is not a store owner) in the requested store | Invaild StoreID, valid product information | Failed authentication, a matching error message will be returned. | Bad |
| The store owner is requesting to remove a non-existent product | Valid StoreID, valid product information | Failed inventory availability, a matching error message will be returned. | Bad |
| The store owner is requesting to add a new product with invalid product information (quantity -1) | Valid StoreID, invalid product information | Failed validity check, a matching error message will be returned. | Bad |
| The store owner is requesting to update an existing product, with valid product information | Valid StoreID, valid product information | Success, the product will be updated in the inventory. | Good |

Use case: **4.2** – **Update store policies** (Update purchase policy, Update discount policy)

1. **Actor**: Store Owner
2. **Pre-condition**: The store owner exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, the store's purchase/discount policy is updated according to the given parameters.
4. **Input parameters**: StoreID, Purchase/Discount policy information.
5. **Actions**:
6. Store Owner attempts to update the purchase/discount policy of a specific shop.
7. Authentication – the system checks that the action was requested by an actual store owner of the needed store.
8. Validity – the system checks that the given parameters are valid (positive,non-empty) and follow the standard rules of the store.
9. The system updates the purchase/discount policy of the store.
10. The system returns a matching success response to the user.
11. **Alternatives/Extensions**:
12. If the authentication failed, a matching error message will be returned and the action will fail.
13. If the given parameters are invalid (for example zero or empty) or do not follow the standard rules of the store, a matching error message will be returned and the action will fail.
14. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The requesting store owner does not have the right permissions (is not a store owner) in the requested store | Invaild StoreID, valid discount policy information | Failed authentication, a matching error message will be returned. | Bad |
| The store owner is requesting to update with invalid discount policy information (quantity -1, discount -300%) | Valid StoreID, invalid discount policy information | Failed validity check, a matching error message will be returned. | Bad |
| The store owner is requesting to update the discount policy of a store, with valid discount policy information | Valid StoreID, valid discount policy information | Success, the discount policy of the store will be updated. | Good |

Use case: **4.3** – **Store Owner Appointment** (Composite: Appointment Offer, Appointment Approval/Rejection)

Use case: **4.3.1** – **Store Owner Appointment Offer**

1. **Actor**: Store Owner, Subscriber
2. **Pre-condition**: The store owner exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, the selected subscriber will receive an appointment offer and will need to accept/reject the offer in the next part of the use case.
4. **Input parameters**: StoreID, username of the subscriber to be appointed as store owner.
5. **Actions**:
6. Store Owner attempts to appoint a subscriber to be a store owner of a specific store.
7. Authentication – the system checks that the action was requested by an actual store owner of the needed store.
8. Validity – the system checks that the given parameters are valid – the given username corresponds to an actual subscriber of the market, the subscriber is not already a store owner of the given store, etc.
9. The system sends an store owner appointment offer to the selected subscriber.
10. The system returns a matching success response to the user.
11. **Alternatives/Extensions**:
12. If the authentication failed, a matching error message will be returned and the action will fail.
13. If the given parameters are invalid (for example if the given username doesn't belong to an actual subscriber, or if the subscriber is already a store owner of the store), a matching error message will be returned and the action will fail.
14. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The requesting store owner does not have the right permissions (is not a store owner) in the requested store | Invaild StoreID, valid subscriber username | Failed authentication, a matching error message will be returned. | Bad |
| The store owner is requesting to appoint a store owner with invalid username (does not belong to a market subscriber) | Valid StoreID, invalid subscriber username | Failed validity check, a matching error message will be returned. | Bad |
| The store owner is requesting to appoint a store owner of a store, with valid subscriber username of a subscriber that isn't already a store owner of the specified store | Valid StoreID, valid subscriber username | Success, the subscriber will receive a store owner appointment offer, which will be accepted/rejected by him in the second part of the appointment use case. | Good |

Use case: **4.3.2** – **Store Owner Appointment Approval/Rejection**

1. **Actor**: Subscriber, Store Owner
2. **Pre-condition**: The subscriber exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, if the subscriber accepted the appointment offer, the selected subscriber will become an owner of the selected store, marking the new owner as appointed by the previous owner.
4. **Input parameters**: The appointment offer id, approval/rejection of the selected subscriber to the given offer.
5. **Actions**:
6. Subscriber is notified about an incoming appointment offer.
7. Subscriber attempts to accept/reject the appointment offer.
8. Authentication – the system checks that the action was requested by a subscriber of the market.
9. Validity – the system checks that the given parameters are valid (appointment offer id actually exists in the system and belongs to the requesting subscriber).
10. If the subscriber accepted the appointment offer, the subscriber will become an owner of the given store, marking the new owner as appointed by the previous owner and the requesting store owner will be notified accordingly. Otherwise, if the offer is rejected, the system dismisses it and notifies the requesting store owner accordingly.
11. The system returns a matching success response to the user.
12. **Alternatives/Extensions**:
13. If the authentication failed, a matching error message will be returned and the action will fail.
14. If the given parameters are invalid (appointment offer id doesn't exist in the system or doesn't belong to the requesting subscriber), a matching error message will be returned and the action will fail.
15. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The user sent an approval request which did not originate from an actual market subscriber | Invaild appointment offer id,  offer approval | Failed authentication, a matching error message will be returned. | Bad |
| The subscriber sent an approval request for an appointment offer that didn't belong to him | Invalid appointment offer id,  offer approval | Failed validity check, a matching error message will be returned. | Bad |
| The subscriber approved an appointment offer that belonged to him | Valid appointment offer id,  offer approval | Success, the subscriber will become an owner of the given store, marking the new owner as appointed by the previous owner and the original requesting store owner will be notified accordingly | Good |

Use case: **4.6** – **Store Manager Appointment** (Composite: Appointment Offer, Appointment Approval/Rejection)

Use case: **4.6.1** – **Store Manager Appointment Offer**

1. **Actor**: Store Owner, Subscriber
2. **Pre-condition**: The store owner exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, the selected subscriber will receive an appointment offer and will need to accept/reject the offer in the next part of the use case.
4. **Input parameters**: StoreID, username of the subscriber to be appointed as store manager.
5. **Actions**:
6. Store Owner attempts to appoint a subscriber to be a store manager of a specific store.
7. Authentication – the system checks that the action was requested by an actual store owner of the needed store.
8. Validity – the system checks that the given parameters are valid – the given username corresponds to an actual subscriber of the market, the subscriber is not already a store owner/store manager of the given store, etc.
9. The system sends an store manager appointment offer to the selected subscriber.
10. The system returns a matching success response to the user.
11. **Alternatives/Extensions**:
12. If the authentication failed, a matching error message will be returned and the action will fail.
13. If the given parameters are invalid (for example if the given username doesn't belong to an actual subscriber, or if the subscriber is already a store owner/store manager of the store), a matching error message will be returned and the action will fail.
14. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The requesting store owner does not have the right permissions (is not a store owner) in the requested store | Invaild StoreID, valid subscriber username | Failed authentication, a matching error message will be returned. | Bad |
| The store owner is requesting to appoint a store manager with invalid username (does not belong to a market subscriber) | Valid StoreID, invalid subscriber username | Failed validity check, a matching error message will be returned. | Bad |
| The store owner is requesting to appoint a store manager of a store, with valid subscriber username of a subscriber that isn't already a store owner or a store manager of the specified store | Valid StoreID, valid subscriber username | Success, the subscriber will receive a store manager appointment offer, which will be accepted/rejected by him in the second part of the appointment use case. | Good |

Use case: **4.6.2** – **Store Manager Appointment Approval/Rejection**

1. **Actor**: Subscriber, Store Owner
2. **Pre-condition**: The subscriber exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, if the subscriber accepted the appointment offer, the selected subscriber will become a manager of the selected store, marking the new manager as appointed by the previous owner.
4. **Input parameters**: The appointment offer id, approval/rejection of the selected subscriber to the given offer.
5. **Actions**:
6. Subscriber is notified about an incoming appointment offer.
7. Subscriber attempts to accept/reject the appointment offer.
8. Authentication – the system checks that the action was requested by a subscriber of the market.
9. Validity – the system checks that the given parameters are valid (appointment offer id actually exists in the system and belongs to the requesting subscriber).
10. If the subscriber accepted the appointment offer, the subscriber will become a manager of the given store, marking the new manager as appointed by the requesting store owner, and the store owner will be notified accordingly. Otherwise, if the offer is rejected, the system dismisses it and notifies the requesting store owner accordingly.
11. The system returns a matching success response to the user.
12. **Alternatives/Extensions**:
13. If the authentication failed, a matching error message will be returned and the action will fail.
14. If the given parameters are invalid (appointment offer id doesn't exist in the system or doesn't belong to the requesting subscriber), a matching error message will be returned and the action will fail.
15. **Acceptance tests – scenarios**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Description** | **Parameters** | **Expected Result** | **Scenario** |
| The user sent an approval request which did not originate from an actual market subscriber | Invaild appointment offer id,  offer approval | Failed authentication, a matching error message will be returned. | Bad |
| The subscriber sent an approval request for an appointment offer that didn't belong to him | Invalid appointment offer id,  offer approval | Failed validity check, a matching error message will be returned. | Bad |
| The subscriber approved an appointment offer that belonged to him | Valid appointment offer id,  offer approval | Success, the subscriber will become a manager of the given store, marking the new manager as appointed by the requesting store owner, and the store owner will be notified accordingly | Good |

Use case: **4.7** – **Store Manager Permission Management**

**Actors:**

Store owner

**Preconditions:**

The store owner exists and logged in and the store exists.

**Postconditions:**

1. The selected manager's permissions have been customized according to the settings defined by the store owner.
2. The manager can now access and perform only the tasks and functions permitted by their customized permissions.

**Input parameters**: StoreID, MangerID, removed permissions, added permissions.

**Main Scenario:**

1. **Authentication** – the system checks that the action was requested by an actual store owner of the needed store.
2. **Selecting a manager –** the store owner chooses a store manager that he has appointed.
3. The system checks that the chosen manager was appointed by the requesting store owner.
4. the store owner add/remove permissions from the selected store manager.
5. The system checks that the permissions are valid
6. The system updates the manager's permissions with the customized permissions.

**Alternatives/Extensions**:

1. If the authentication failed, a matching error message will be returned and the action will fail.
2. If the chosen store manager wanst appointed by the store owner, a matching error message will be returned and the action will fail
3. If the permissions are invalid (unrecognized to the system/ can't be assigned to store manager, a matching error message will be returned and the action will fail

**Test 1: Successful Customization of Manager Permissions**

**Input**: StoreID, ManagerID, removed permissions, added permissions.

**Preconditions**: The store owner is logged in and exists.

**Steps**:

1. Store owner selects a manager for permission customization.

2. Removed and added permissions are specified.

3. Permissions are valid and can be assigned to the manager.

**Expected Outcome:**

* The selected manager’s permissions are updated according to the provided settings.
* The manager can access and perform only the tasks and functions permitted by their customized permissions.

**Test 2: Authentication Failure**

**Input:** StoreID, ManagerID, removed permissions, added permissions.

**Preconditions:** The store owner is not logged in or not exist.

**Steps:**

Store owner attempts to customize manager permissions.

**Expected Outcome:**

* Error message
* The action fails, and no changes are made to the manager’s permissions.

**Test 3: Invalid Manager Selection**

**Input:** StoreID, ManagerID, removed permissions, added permissions.

**Preconditions:**

The store owner is logged in and is an owner of the store.

**Steps:**

Store owner selects a manager that was not appointed by them.

**Expected Outcome:**

* Error message
* The action fails, and no changes are made to the manager’s permissions.

**Test 4: Invalid Permissions**

**Input:** StoreID, ManagerID, removed permissions, added permissions.

**Preconditions:**

The store owner is logged in and is an owner of the store.

**Steps:**

Store owner attempts to assign invalid or unrecognized permissions to the manager.

**Expected Outcome:**

* Error message
* The action fails, and no changes are made to the manager’s permissions.

**Test 5: Permissions Conflict**

**Input:** StoreID, ManagerID, removed permissions, added permissions.

**Preconditions:**

The store owner is logged in and is an owner of the store.

**Steps:**

Store owner attempts to assign permissions that conflict with existing permissions of the manager (adding an existing permission for example).

**Expected Outcome:**

* Error message
* The action fails, and no changes are made to the manager’s permissions.

Use case: **4.9** – **Store Manager Permission Management**

**Actors:**

Store founder.

**Preconditions:**

The store founder exists and logged in and the store exists.

**Postconditions:**

1. The store is closed and becomes inactive.
2. Access to the closed store’s information is restricted to store owners and system administrators.
3. Store owners and managers receive a notification about the closure, but their appointment remains unchanged.
4. Store’s products will no longer apear while searching for products.

**Input parameters**: StoreID

**Main Scenario:**

1. **Authentication** – the system checks that the action was requested by the store founder of the needed store.
2. The system checks that the store is currently open
3. The system change the store status to inactive
4. The system sends a notification informing the store owners and managers of the store closure.

**Alternatives/Extensions**:

1. If the authentication failed, a matching error message will be returned and the action will fail.
2. If the store is allready closed, a matching error message will be returned and the action will fail.

**Test 1: Successful Closure of the Store**

**Input:** StoreID

**Preconditions:** The store founder is logged in and is the founder of the store.

**Steps:**

Store founder initiates the closure of the store.

**Expected Outcome:**

* The store status is changed to inactive.
* Access to the closed store’s information is restricted to store owners and system administrators.
* Store owners and managers receive a notification about the closure, but their appointment remains unchanged.
* Store’s products will no longer appear while searching for products.

**Test 2: Authentication Failure**

**Input:** StoreID

**Preconditions:**

The store founder is not logged in or is not the founder of the store.

**Steps:**

Unauthorized user attempts to close the store.

**Expected Outcome:**

* Error message
* The action fails, and the store status remains unchanged.

**Test 3: Attempting to Close an Already Closed Store**

**Input:** StoreID

**Preconditions:**

The store founder is logged in and is an owner of the store.

The store is already closed.

**Steps:**

* 1. Store founder closes the store.
  2. Store founder attempts to close the store again.

**Expected Outcome:**

* Error message
* The action fails, and the store status remains unchanged.

Use case: **4.11** – **Request for Information on Store Roles**

**Actors:**

Store owner.

**Preconditions:**

The store owner exists and logged in and the store exists.

**Postconditions:**

The store owner has access to detailed information about the roles in their store.

**Input parameters**: StoreID

**Main Scenario:**

1. **Authentication** – the system checks that the action was requested by a store owner of the needed store.
2. The system retrieve inforamtion about all the roles in the store.

**Alternatives/Extensions**:

If the authentication failed, a matching error message will be returned and the action will fail.

**Test 1: Successful Retrieval of Store Roles Information**

**Input:** StoreID

**Preconditions:**

The store owner is logged in and is an owner of the store.

**Steps:**

Store owner initiates a request for information on store roles.

**Expected Outcome:**

The system retrieves detailed information about all the roles in the store.

**Test 2: Authentication Failure**

**Input:** StoreID

**Preconditions:**

The store owner is not logged in or is not an owner of the store.

**Steps:**

Unauthorized user attempts to request information on store roles.

**Expected Outcome:**

* Error message
* The action fails, and the requested information is not retrieved.

Use case: **4.11** – **Request for Information on manager permissions.**

**Actors:**

Store owner.

**Preconditions:**

The store owner exists and logged in and the store exists.

**Postconditions:**

The store owner has access to detailed information about the roles in their store.

**Input parameters**: StoreID, ManagerID

**Main Scenario:**

1. **Authentication** – the system checks that the action was requested by a store owner of the needed store.
2. The system checks that the manager exists and is a manager in the store
3. The system retrieve the manager permissions in the store.

**Alternatives/Extensions**:

1. If the authentication failed, a matching error message will be returned and the action will fail.
2. If the manager doesn't exists or is not a manager of the store, a matching error message will be returned and the action will fail.

**Test 1: Successful Retrieval of Manager Permissions**

**Input:** StoreID, ManagerID

**Preconditions:** The store owner is logged in and is an owner of the store. The manager exists and is assigned as a manager in the store.

**Steps:**

Store owner initiates a request for information on manager permissions.

**Expected Outcome:**

* The system retrieves information about the permissions assigned to the specified manager in the store.

**Test 2: Authentication Failure**

**Input:** StoreID, ManagerID

**Preconditions:**

The store owner is not logged in or not an owner of the store.

**Steps:**

Unauthorized user attempts to request information on manager permissions.

**Expected Outcome:**

* Error message
* The action fails, and the requested information is not retrieved.

**Test 3: Manager Does Not Exist or Is Not a Manager in the Store**

**Input:** StoreID, ManagerID

**Preconditions:** The store owner is logged in and is an owner of the store.

**Steps:**

* 1. Store owner initiates a request for information on manager permissions.
  2. Store owner specifies a ManagerID that does not exist or is not assigned as a manager in the store.

**Expected Outcome:**

* Error message
* The action fails, and the requested information is not retrieved.

Use case: **4.13** – **Purchase History Retrieval.**

**Actors:**

Store owner.

**Preconditions:**

The store owner exists and logged in and the store exists.

**Postconditions:**

The store owner has access to detailed purchase history information.

**Input parameters**: StoreID

**Main Scenario:**

1. **Authentication** – the system checks that the action was requested by a store owner of the needed store.
2. The system retrieves the purchase history of the store.

**Alternatives/Extensions**:

1. If the authentication failed, a matching error message will be returned and the action will fail.
2. If there were no purchases from the store, a message telling there was no purchases will be returned.

**Test 1: Successful Retrieval of Purchase History**

**Input:** StoreID

**Preconditions:**

The store owner is logged in and is an owner of the store.

The store exists.

**Steps:**

Store owner initiates a request to retrieve purchase history.

**Expected Outcome:**

* The system retrieves purchase history for the specified store.

**Test 2: Authentication Failure**

**Input:** StoreID

**Preconditions:**

The store owner is not logged in or not an owner of the store.

**Steps:**

Unauthorized user attempts to retrieve purchase history.

**Expected Outcome:**

* Error message
* The action fails, and the requested information is not retrieved.

**Test 3: No Purchase History Available**

**Input:** StoreID

**Preconditions:**

The store owner is logged in and is an owner of the store.

The store exists

**Steps:**

Store owner initiates a request to retrieve purchase history.

**Expected Outcome:**

* Error message
* The action fails, and the requested information is not retrieved.

Use case: **5** –**store manager permitted actions.**

**Actors:**

Store manager.

**Preconditions:**

The store manager exists and logged in and the store exists.

**Postconditions:**

The requested action is performed.

**Input parameters**: StoreID, action.

**Main Scenario:**

1. **Authentication** – the system checks that the action was requested by a store manager of the needed store.
2. The system checks if the action is within his permissions.
3. The system perform that action

**Alternatives/Extensions**:

1. If the authentication failed, a matching error message will be returned and the action will fail.
2. If the action is not within the store manager permissions, a matching error message will be returned and the action will fail.

**Test 1: Successful Execution of Permitted Action**

**Input:** StoreID, action

**Preconditions:**

The store manager is logged in and is a manager of the store.

The store exists.

**Steps:**

Store manager initiates the requested action.

**Expected Outcome:**

The system successfully performs the requested action

**Test 2: Authentication Failure**

**Input:** StoreID, action

**Preconditions:**

The store manager is not logged in or not a manager of the store.

**Steps:**

Unauthorized user attempts to initiate the requested action.

**Expected Outcome:**

* Error message
* The action fails, and the requested information is not retrieved.

**Test 3: Action Not Within Store Manager Permissions**

**Input:** StoreID, action

**Preconditions:**

The store manager is logged in and is a manager of the store.

The store exists.

The requested action is not within the store manager's permissions.

**Steps:**

Store manager attempts to initiate the requested action.

**Expected Outcome:**

* Error message
* The action fails, and the requested information is not retrieved.