Use cases – v1

Table of Contents

System use cases:

1.1 – Initialization of the system:

A user opens the system legally. The system manager is this user.

Actors:

- User (System Manager)

Precondition:

- The system is off and being activated for the first time.

Parameters:

- Correct username and password for the system
- Payment and supply services details

Postcondition:

- The system is on and has a system manager (the user who activated it), and initialized payment and supply services.

Main Scenario:

1. User turns on the system.
2. System asks the user to insert his username and password.
3. User enter his authentication details.
4. System receives correct username and password from the user and verifies that they are correct.
5. System activates the system and initializes it with one member which is also the system manager, and with supply and payment services.
6. System reports a successful activation of the system to the user who initialized it.
7. System then starts to serve users.

Alternative Scenarios:

1. User enters an invalid username or password at step 3, the system reports the error and prompts the user to re-enter them.
2. Payment service or supply service details are incorrect at step 5, the system reports the error and prompts the user to correct the details.

1.2 – Changing/Replacing/Adding Contact with External Services:

The system manager changes/replaces/adds an external service of the system.

Actors:

- System manager

Precondition:

- The system is on and initialized correctly. The system manager is logged in.

Parameters:

- Details about the external service to be changed/replaced/added.

Postcondition:

- External services are updated without changing the system activity.

Main Scenario:

1. System manager selects the option to manage external services.
2. System manager selects the service to be changed/replaced/added.
3. System manager updates the details for the selected service.
4. System verifies the validity of the changes.
5. System updates the external service contact details.
6. System confirms successful update to the system manager.

Alternative Scenarios:

- Service updated details are incorrect in step 4, the system manager asks the user to enter them again correctly.

1.3 – Payment:

The system contacts the payment service which the system is familiar with to make a payment and receive an approvement of the payment.

Actors:

- System

Precondition:

- The system is on.

Parameters:

- Payment details

Postcondition:

- User receives confirmation of successful payment.

Main Scenario:

1. System contacts with the payment service with payment details.
2. External payment service processes the details and confirms successful payment.

Alternative Scenarios:

- The external payment service reports of unsuccessful payment in step 2.
- The payment details sent by the system in step 1 to the external payment service are wrong.

1.4 – Supply:

The system contacts the supply service which the system is familiar with in to make a supplement order and receive an approvement of the order.

Actors:

- System

Precondition:

- The system is on.

Parameters:

- Supplement details.

Postcondition:

- System receives confirmation of successful payment.

Main Scenario:

1. System contacts with the supply service with supplement details and client details.
2. External supply service processes the details and sends back answer of successful supplement.

Alternative Scenarios:

- The external supply service reports of unsuccessful supplement request in step 2.
- The supplement details sent by the system in step 1 to the external supply service are wrong.

1.5 – Real-time Alerts:

For store owner-

System should send a real time notification to logged in store owners when a client has bought one of their products, when one of their stores is closed\reopened, or when their subscription as a store owner is removed.

Actors:

- Store Owner

Precondition:

- System is on.
- Store owner is logged in.

Parameters:

- Type of notification

Postcondition:

- Store owner receives a real time notification.

Main Scenario:

1. One of the 4 scenarios happens: a client has bought one of their products, when one of their stores is closed\reopened, or when their subscription is removed.
2. System notifies the real time alert to the store owner.

Alternative Scenarios:

- alert delivery fails in step 2, the system sends it again.

For member-

System should send a real time notification to logged in store members when he receives a message.

Actors:

- Member

Precondition:

- System is on.
- Member is logged in.

Parameters:

- Type of notification

Postcondition:

- Member receives a real time notification.

Main Scenario:

1. The member receives message.
2. System notifies the real time alert to the member.

Alternative Scenarios:

- alert delivery fails in step 2, the system sends it again.

1.6– Delayed Alerts:

Actors:

- Member

Precondition:

- System is on.
- Member is logged in.

Parameters:

- Pending alerts.

Postcondition:

- Member receives pending alerts upon logging into the market.

Main Scenario:

1. Member logs into the system.
2. System presents pending alerts to the member.

Alternative Scenarios:

- member has no pending alerts in step 2, the system does not present any alerts and presents the message: "no new notifications".

Guest use cases:

2.1.1 – Guest Login:

User can enter the system as a guest. The system assigns him a private shopping cart.

Actors:

- User

Precondition:

- The user is not logged into the system.

Parameters:

None

Postcondition:

- The user is connected to the system as a guest and receives a private shopping cart.

Main Scenario:

1. User accesses the system.
2. System defines the user as a guest and creates a private shopping cart for him.
3. User receives a shopping cart and can function as a guest in the system.

Alternative Scenarios:

- System fails the connection at any step (1-3), it cancels it's actions.

2.1.2 – Exit:

Guest can disconnect the system, and his shopping cart removed by the system.

Actors:

- User

Precondition:

- The guest is connected to the system.

Parameters:

None

Postcondition:

- The user is logged out of the system as a guest and loses his shopping cart.

Main Scenario:

1. Guest selects the option to log out of the system.
2. System deletes his shopping cart and closes the connection with him.
3. Guest loses their shopping cart and is no longer identified as a guest.

Alternative Scenarios:

- The user closes the connection before step 2, the system just deletes the shopping cart and acts like the request succeeded.

2.1.3- Registration to the system:

A guest can register by providing unique identification details. At the end of successful registration process, the guest is registered in the system as a member. Still, to get member status, he must log in with the details.

Actors:

- User

Precondition:

- The user is connected to the system as a guest.

Parameters:

- Authentication details.

Postcondition:

- The member is registered in the system and can log in with the authentication details.

Main Scenario:

1. Guest accesses the registration page of the system and enters his authentication details.
2. System verifies the details and their uniqueness.
3. System registers the user as a member.
4. User is registered in the system as a member.
5. System alerts the user that the action succeeded.

Alternative Scenarios:

- Identification details are not unique in step 2, the system prompts the user to provide different details.

2.1.4 – Member Login:

A guest who registered the system as a member, can now login to the system as a member using his authentication details.

Actors:

- User

Precondition:

- The user is registered in the system but not logged in.

Parameters:

- Authentication details.

Postcondition:

- The user is logged into the system as a member.

Main Scenario:

1. Guest requests to log in the system and enters his identifying details.
2. System verifies the guest's identifying details.
3. System logs the user into the system as a member and changes it's state.
4. System alerts the user that the action succeeded.

Alternative Scenarios:

- Identifying details are incorrect in step 2, the system prompts the user to re-enter the details.

2.2.1 – Receiving Information about stores and products:

A guest can access information about stores and their products.

Actors:

- User

Precondition:

- user is logged into the system as a guest.

Parameters:

- none

Postcondition:

- User receives information about stores and their products in the system.

Main Scenario:

1. User requests information about the stores and their products in the system.
2. System returns a list of stores available in the market.
3. User selects a specific store to view its products.
4. System retrieves and displays the products available in the selected store.
5. System alerts the user the action as succeeded.

Alternative Scenarios:

- In step 3 there are no stores, the system displays a message : "no active stores in the market" and ends the action.

2.2.2 – Product Search:

Guest can search products without focusing on a specific store, by product name, category or keywords, and also filter the results according to characteristics such as: price range, product rating, category, store rating, etc.

Actors:

- User

Precondition:

- User is logged into the system as a guest.

Parameters:

- Search fields.

Postcondition:

- Guest receives search results based on specified criteria.

Main Scenario (a):

1. User accesses the search functionality within the system.
2. User enters search criteria such as product name, category, or keywords.
3. System retrieves products matching the search criteria.
4. User applies filters such as price range, product rating, category, and store rating.
5. System displays filtered search results to the user.

Main Scenario (b):

1. User navigates to a specific store within the system.
2. User enters search criteria for products available in the store.
3. System retrieves products matching the search criteria.
4. User applies filters as described in scenario (a).
5. System displays filtered search results to the user.

Alternative Scenarios:

- There are no results for the search in steps a3, b3, system displays "no matching results".

2.2.3 – Adding products to shopping cart as a guest:

Guest can add products to his shopping cart for each store in the system. Each store has a basket in the shopping cart.

Actors:

- User

Precondition:

- User is logged into the system as a guest.

Parameters:

- Store id, product id, quantity.

Postcondition:

- Products are saved in the user's shopping cart in the basket of store where the product is bought from.

Main Scenario:

1. Guest selects products and their amounts from stores in the system to add to the shopping cart.
2. System verifies that the parameters are correct and exists in the system and in the store stock, and adds the selected products to the guest's shopping cart to the appropriate baskets.
3. System alerts the user that his action succeeded.

Alternative Scenarios:

- Verification in step 2 failed, the system alerts the guest about unsuccessful operation.

2.2.4 – Checking and Modifying Shopping Cart:

A guest can watch his shopping cart and make changes in it – update amount of products in cart.

Actors:

- User

Precondition:

- User is logged into the system as a guest.

Parameters:

- Store id, product id, amount.

Postcondition:

- The guest's shopping cart contains the specified amount of products in the specified store's basket.

Main Scenario:

1. Guest accesses the shopping cart within the system.
2. System verifies that a basket of the store id entered exists in the shopping cart, amount is a positive integer number, and that product id appears in the store id basket.
3. System displays the contents of the shopping cart to the user.
4. Guest reviews the products in the shopping cart and their quantities.
5. Guest updates product quantities.
6. System updates the shopping cart according to the user's modifications.
7. System alerts the user that the action succeeded.

Alternative Scenarios:

- Verification in step 2 failed- system alerts the user that the action has failed.

2.2.5 – Purchase of Shopping Cart:

2.2.5.a– Make purchase of all the products in cart:

A guest can make purchase of his shopping cart if all the products in cart are available in stock of their stores.

Actors:

- User

Precondition:

- All products in the shopping cart are available in store stock.

Postcondition:

- Guest completes the purchase of the entire shopping cart, and the stock quantity of each product is updated.
- Purchase is updated in the purchase history.

Main Scenario:

1. Guest sends a purchase request.
2. System verifies the availability of the products in stock.
3. System checks the purchase constraints and verifies that this purchase is legal and meets all the constraints in this purchase.
4. System verifies delivery service.
5. System applies purchase policies and discounts to the items in the shopping cart.
6. System verifies payment service.
7. System proceeds with the purchase of the entire shopping cart.
8. System adds new purchase details to the purchases history.
9. System empties the cart of the user.
10. System sends a response to the user with order approval.

Alternative Scenarios:

- Payment service fails verification in step 6, the system cancels the action and sends an error message that action failed.
- Delivery service fails verification in step 4, the system cancels the action and sends an error message that action failed.
- Purchase is not legal int step 3, the system cancels the action and sends an error message with details of the unfulfilled constraints.

2.2.5.b – Cancel Purchase of Entire Shopping Cart:

Shopping cart cannot be purchased in its entirety due to unavailable products, it is required not to make the purchase at all ("all or nothing").

Actors:

- User

Precondition:

- Required amount of some products in the shopping cart is unavailable in store stock.

Postcondition:

- Purchase of the entire shopping cart is cancelled.

Main Scenario:

1. Guest sends a purchase request.
2. System verifies the availability of the products in stock.
3. System cancels the purchase of the entire shopping cart.
4. System sends an error response to user with details of the products that are not available.

Alternative Scenarios:

- None

Member use cases:

2.3.1 – Member Logout:

When a member logs out of the system, he becomes a guest again.

Actors:

- Member

Precondition:

- User is logged into the system as a member.

Parameters:

- none

Postcondition:

- User's identification is cancelled, and he became a guest again.
- Member's shopping cart is not deleted.

Main Scenario:

1. Member selects the option to logout from the system.
2. System saves the member's shopping cart.
3. System logs the member out of the system and changes his state to a guest.
4. User's identification is cancelled, reverting him back to a guest status, his shopping cart is not deleted and will be saved for him in the next login.
5. System sends a successful response.

Alternative Scenarios:

- Member closes the connection without request to logout in step 1, system acts like a logout request has been received.

2.3.2 – Opening a new store:

A member logged in the system can open a store and become the owner of the store- the first store owner.

Actors:

- Member

Precondition:

- User is logged into the system as a member.

Parameters:

- Store details.

Postcondition:

- User becomes the owner of a new store in the system.

Main Scenario:

1. Member selects the option to open a new store.
2. Member provides necessary details for the new store, such as store name, description, and category.
3. System verifies the provided details.
4. System creates the new store and designates the user as the first owner of it.
5. Member becomes the owner of the new store in the system.
6. System sends a successful response with new store id.

Alternative Scenarios:

- Details provided by the member are incomplete or incorrect in step 3, the system prompts the user to correct them before proceeding with store creation.

Store Owner use cases:

2.4.1 – Stock Management:

2.4.1.a – Adding new product to the store:

The store owner can add product to the store.

Actors:

- Store owner

Precondition:

- Store is active
- Member is logged into the trading system as a store owner.

Parameters:

- Product details

Postcondition:

- System continues to work usually, with the new product in given store's catalog.

Main Scenario:

1. owner sends request for adding new product to the store.
2. System checks that the product details are valid.
3. System adds the product to the store collection with the given details and initial quantity of 0.

Alternative Scenarios:

- The parameters that were given are not valid details (found in step 2)- the system returns appropriate error message with the specific information.

2.4.1.b – Removing products from the store:

The store owner can remove products from the store.

Actors:

- Store owner

Precondition:

- Store is active
- Member is logged into the trading system as a store owner
- Product id is of an active product in the store.

Parameters:

- Product id

Postcondition:

- System continues to work usually, without the product in given store's catalog.

Main Scenario:

1. owner sends request for removing existing product from the store.
2. System checks that the product id is valid.
3. System removes the product from the store collection.
4. System sends a response that indicates the removal of the product.

Alternative Scenarios:

- The parameter that was given is not a valid id (the product does not exist- found in step 2)- the system returns appropriate error message with the specific information.

2.4.1.c – Updating product's quantity in stock:

The store owner can change the quantity of a product from the store.

Actors:

- Store owner

Precondition:

- Store is active
- Member is logged into the trading system as a store owner
- Product id is of an active product in the store.

Parameters:

- Product id
- New quantity

Postcondition:

- System continues to work usually, with the product quantity equals to the given quantity.

Main Scenario:

1. owner sends request for updating product's quantity in the store.
2. System checks that the product id and quantity are valid.
3. System updates the product's quantity in the store stock.
4. System sends a response that indicates the quantity was updated.

Alternative Scenarios:

- The parameter that was given is not a valid id (the product does not exist- found in step 2)- the system returns appropriate error message with the specific information.

- The parameter that was given is not a valid quantity (negative- found in step 2)- the system returns appropriate error message with the specific information.

2.4.1.d – Updating product's details in given store:

The store owner can change the details of a product from the store.

Actors:

- Store owner

Precondition:

- Store is active
- Member is logged into the trading system as a store owner
- Product id is of an active product in the store.

Parameters:

- Product id
- Product details

Postcondition:

- System continues to work usually, with the product details equal to the given details.

Main Scenario:

1. owner sends request for updating product's details in the store.
2. System checks that the product id and details are valid.
3. System updates the product's details in the store.
4. System sends a response that indicates the details were updated.

Alternative Scenarios:

- The parameter that was given is not a valid id (the product does not exist- found in step 2)- the system returns appropriate error message with the specific information.
- The parameter that was given is not valid details (found in step 2)- the system returns appropriate error message with the specific information.

2.4.2 – Update type of purchases and discount policy:

The store owner can change the purchases and discount policy of the store.

Actors:

- Store owner

Precondition:

- Store is active
- Member is logged into the trading system as a store owner

Parameters:

- Store id
- Product id
- Purchases and discount policy

Postcondition:

- New policy is applied.

Main Scenario:

1. owner sends request for updating store's policy.
2. System checks that product id is valid (product exists in the store).
2. System sets the new policy for the store.
3. System sends a response that indicates the policy was updated.

Alternative Scenarios:

- The parameter that was given is not a valid id (the product does not exist- found in step 2)- the system returns appropriate error message with the specific information.

2.4.3 – Appoint store owner:

The store owner can appoint another member to be a store owner.

Actors:

- Store owner

Precondition:

- Store is active
- Member is logged into the trading system as a store owner

Parameters:

- Member id

Postcondition:

- Store owners hierarchy has no cycles (tree)

Main Scenario:

1. Owner sends request for appointing new store owner.
2. System verifies the member exists and is not already an owner.
3. System sends the given member an appointment offer.
4. Member (given) accepts the offer.
5. System sets the new owner under the actor in the hierarchy.
6. System sends a response that indicates the new owner was appointed.

Alternative Scenarios:

- The member does not exist (found in step 2)- the system cancels the action.
- The member is already an owner of the store (found in step 2)- the system cancels the action.
- The member (given) denies the appointment offer (found in step 3)- the system cancels the action and returns an appropriate message with the specific information.

2.4.6 – Appoint store manager:

The store owner can appoint another member to be a store manager.

Actors:

- Store owner

Precondition:

- Store is active
- Member is logged into the trading system as a store owner

Parameters:

- Member id
- Manager permissions

Postcondition:

- The member is assigned to be a manager with the given permissions and the actor as the appointer.

Main Scenario:

1. Owner sends request for appointing new store manager.
2. System verifies the member exists, is not already a manager and the permissions are legal.
3. System sends the given member an appointment offer.
4. Member (given) accepts the offer.
5. System sets the new manager with the actor as appointer.
6. System sends a response that indicates the new manager was appointed.

Alternative Scenarios:

- The member does not exist (found in step 2)- the system cancels the action.
- The member is already a manager of the store (found in step 2)- the system cancels the action.
- The permissions are not legal (found in step 2)- the system cancels the action.
- The member (given) denies the appointment offer (found in step 3)- the system cancels the action and returns an appropriate message with the specific information.

2.4.7 – Update store manager permissions:

The store owner can edit each manager's permissions.

Actors:

- Store owner

Precondition:

- Store is active
- Member is logged into the trading system as a store owner
- Store Owner (actor) is the appointer of the given manager

Parameters:

- Member id
- New manager permissions

Postcondition:

- Manager's new permissions are updated.

Main Scenario:

1. Owner sends request to edit store manager's permissions.
2. System verifies the given member exists and is a manager in this store.
3. System verifies that the manager was appointed by the actor.
4. System updates manager permissions respectively.
5. System sends a response that indicates the permissions were updated.

Alternative Scenarios:

- The member does not exist (found in step 2)- the system cancels the action and returns an appropriate message with the specific information.
- The member is not a manager of the store (found in step 2)- the system cancels the action and returns an appropriate message with the specific information.
- The manager was not appointed by the actor (found in step 3)- the system cancels the action and returns an appropriate message with the specific information.

2.4.9 – Close Store:

The store founder can set the store inactive.

Actors:

- Store founder (first store owner)

Precondition:

- Store is active
- Member is logged into the trading system as a store owner

Parameters:

- Founder id
- Store id

Postcondition:

- Given store's products will not appear in the search results
- The store owners and managers are notified about the close action

Main Scenario:

1. Founder sends request to close the given store.
2. System verifies that the actor is indeed the first owner in the store hierarchy.
3. System verifies that the store is active.
4. System sets the store inactive.
5. System notifies the store owners and managers that the store is closed.
6. System sends a response that indicates the store has been closed.

Alternative Scenarios:

- The actor is not the founder (found in step 2)- the system cancels the action and returns an appropriate message with the specific information.
- The store is inactive (found in step 3)- the system cancels the action and returns an appropriate message with the specific information.

2.4.11 – Get Store Information:

2.4.11.a – Get role holders details:

The store owner can view information of the role holders in the store.

Actors:

- Store owner

Precondition:

- Member is logged into the trading system as a store owner.

Parameters:

- Store owner id
- Store id

Postcondition:

- None

Main Scenario:

1. owner sends request to get information about role holders in the store.
2. System verifies that the store exists in the system.
3. System verifies that the actor is a store owner.
4. System returns a list containing the store role holders data.

Alternative Scenarios:

- The store is not exist (found in step 2)- the system returns an appropriate message with the specific information.
- The actor is not an owner of the store (found in step 3)- the system returns an appropriate message with the specific information.

2.4.11.b – Get managers permissions:

The store owner can view the permissions of the store managers.

Actors:

- Store owner

Precondition:

- Member is logged into the trading system as a store owner.

Parameters:

- Store owner id
- Store id

Postcondition:

- None

Main Scenario:

1. owner sends request to get the store managers permissions.
2. System verifies that the store exists in the system.
3. System verifies that the actor is a store owner.
4. System returns a list containing the store managers permissions.

Alternative Scenarios:

- The store is not exist (found in step 2)- the system returns an appropriate message with the specific information.
- The actor is not an owner of the store (found in step 3)- the system returns an appropriate message with the specific information.

2.4.13 – Get purchase history:

The store owner can view the purchase history in his stores.

Actors:

- Store owner

Precondition:

- Member is logged into the trading system as a store owner.

Parameters:

- Store owner id
- Store id

Postcondition:

- None

Main Scenario:

1. owner sends request to get the store purchase history.
2. System verifies that the store exists in the system.
3. System verifies that the actor is a store owner.
4. System returns a list containing the store purchase history.

Alternative Scenarios:

- The store is not exist (found in step 2)- the system returns an appropriate message with the specific information.
- The actor is not an owner of the store (found in step 3)- the system returns an appropriate message with the specific information.

System Manager use cases:

2.6.4 – Get purchase history of store/buyer by system manager:

The system manager can view the purchase history of a store/buyer.

Actors:

- System manager

Precondition:

- Member is logged into the trading system as a system manager.

Parameters:

- System manager id
- Store/buyer id

Postcondition:

- None

Main Scenario:

1. system manager sends request to get the store/buyer purchase history.
2. System verifies that the store/buyer exists in the system.
3. System verifies that the actor is a system manager.
4. System returns a list containing the store/buyer purchase history.

Alternative Scenarios:

- The store/buyer is not exist (found in step 2)- the system returns an appropriate message with the specific information.
- The actor is not a system manager (found in step 3)- the system returns an appropriate message with the specific information.