Trading system’s use cases

**INDEX:**

Adding item to the shop’s stock (II.4.1.1) 12

Change item’s details in shop’s stock (II.4.1.3) 13

Change/replace/add association with external service (I.2) 2

Deleting item from shop’s stock (II.4.1.2) 12

Discount policy applying for guest (II.2.5.2.2) 10

Discount policy applying for member (II.2.5.2.1) 9

Discount policy changing (II.4.2.2) 14

General items searching (II.2.2) 6

Get into the Trading system as a guest (II.1) 4

info about shop in the market and item in shop requesting (II.2.1) 6

Initialization of the system (I.1) 2

In-shop purchases history request (II.4.13)/( (II.6.4) 18

Items saving in the shopping bag (II.2.3) 7

Login to the trading system (II.1.4) 5

Logout (II.3.1) 11

Payment execution (I.3) 2

Pending notification (I.6) 4

Purchase archiving (II.2.5.3) 10

purchase policy changing (II.4.2.1) 13

Purchase policy verifying (II.2.5.1) 9

Real-time notification (I.5) 3

Registration for the trading system (II.3) 5

Shipping request (I.4) 3

Shop closing (II.4.9) 16

Shop manager assignment (II.4.6) 15

Shop manager permissions updating (II.4.7) 16

Shop opining (II.3.2) 11

Shop owner assignment (II.4.4) 15

Shop’s roles info request (II.4.11) 17

Shopping cart purchase for guest (II.2.5) 7

Shopping cart purchase for Member (II.2.5 \*) 8

Shopping carts content checking and editing (II.2.4) 7

Trading system quitting as a guest (II.2) 4

Trading system quitting as a member (II.1.2\*) 11

# System use cases

|  |  |
| --- | --- |
| **Title:** | Initialization of the system (I.1) |
| **Actors:** | System admin. |
| **Preconditions:** | None |
| **Parameters:** | External services. |
| **Postconditions:** | None |
| **Expected result:** | The system is initialized and loaded; the system is ready to receive users. |
| **Actions:** | 1. The system admin open and initialize the system. 2. System gets up and be available. |

|  |  |
| --- | --- |
| **Title:** | Change/replace/add association with external service (I.2) |
| **Actors:** | System admin. |
| **Preconditions:** | External service exists and functional. |
| **Parameters:** | External service configuration. |
| **Postconditions:** | the system does not get interrupted and corrupted. |
| **Expected result:** | The external service is connected to the system without any defects for market’s activities. |
| **Actions:** | 1. The system admin integrates and connect (add/change/replace) external service to the system. 2. The system approves the admin’s changes while continuing its activities as usual. |

|  |  |
| --- | --- |
| **Title:** | Payment execution (I.3) |
| **Actors:** | User. |
| **Preconditions:** | The user agreed to payment conditions. |
| **Parameters:** | Deal’s details. |
| **Postconditions:** | If the request denied so the purchase could not be done. |
| **Expected result:** | Payment result must be either approval or denied. |
| **Actions:** | 1. The user is completing his purchase and proceed a payment. 2. The system is sending a payment request to the external payment service. 3. The external payment service sends back a result which could be either an approval or deny, the trading system reforwards the relevant details to the user. |

|  |  |
| --- | --- |
| **Title:** | Shipping request (I.4) |
| **Actors:** | 1. Shop owner. 2. Shop manager. |
| **Preconditions:** | 1. If the actor is a shop manager, the actor must have the permission. 2. The supplier must be recognized by the trading system. |
| **Parameters:** | 1. Details of the package and customer. 2. The delivery details. |
| **Postconditions:** | None |
| **Expected result:** | Approval or denied result. |
| **Actions:** | 1. The shop owner or manager (with relevant permission) asks the shipping service for a package with specific details. 2. The system is sending a shipping request to the external shipping service. 3. The external shipping service sends back a result which could be either an approval or deny, the trading system reforwards the relevant details to the shop’s manager/owner (requester). |

|  |  |
| --- | --- |
| **Title:** | Real-time notification (I.5) |
| **Actors:** | Market |
| **Preconditions:** | The member must be active. |
| **Parameters:** | Member details. |
| **Postconditions:** | None |
| **Expected result:** | The member is notified at real-time by the trading system. |
| **Actions:** | 1. Some activity occurs and the member should know during his connection time. 2. The trading system notifies the member immediately at real-time. |

|  |  |
| --- | --- |
| **Title:** | Pending notification (I.6) |
| **Actors:** | Market. |
| **Preconditions:** | Member is not online. |
| **Parameters:** | Member details. |
| **Postconditions:** | None |
| **Expected result:** | The member would get notified at first connection in the future. |
| **Actions:** | 1. Some activity occurs which member should know about. 2. Member is not online. 3. The trading system remembers the member and the notification. 4. When member connects in future, the trading system would notify the member. |

# User use cases

|  |  |
| --- | --- |
| **Title:** | Get into the Trading system as a guest (II.1) |
| **Actors:** | User |
| **Preconditions:** | User is not logged in. |
| **Parameters:** | None |
| **Postconditions:** | 1. The shopping cart must be empty. |
| **Expected result:** | 1. The user gets a shopping cart. 2. user can buy. |
| **Actions:** | 1. user opens the trading system. 2. The system presents a page to start trading as a guest. |

|  |  |
| --- | --- |
| **Title:** | Trading system quitting as a guest (II.2) |
| **Actors:** | Guest |
| **Preconditions:** | 1. User is not logged in. |
| **Parameters:** | None |
| **Postconditions:** | None |
| **Expected result:** | The system removes the guest from the active guests and delete his shopping cart. |
| **Actions:** | 1. The user quits the trading system 2. the trading system clears the user’s shopping cart. 3. The trading system undefines the user as a guest. |

|  |  |
| --- | --- |
| **Title:** | Registration for the trading system (II.3) |
| **Actors:** | Guest |
| **Preconditions:** | None. |
| **Parameters:** | Primary details. |
| **Postconditions:** | 1. The guest must log in in order to trade as a member. |
| **Expected result:** | 1. The trading system registers the guest as a member. |
| **Actions:** | 1. The guest asks to register for the trading system. 2. The system asks the guest to enter primary details. 3. The guest enters primary details. 4. The system registers the guest as a member in the trading system. |

|  |  |
| --- | --- |
| **Title:** | Login to the trading system (II.1.4) |
| **Actors:** | User. |
| **Preconditions:** | User is a guest. |
| **Parameters:** | credentials. |
| **Postconditions:** | If the login was successful, the trading system recognize the user as a member.  Otherwise, the system is showing incorrect credentials error message for the user and asking him to re-enter correct credentials or register to the market. |
| **Expected result:** | If the login was successful, the trading system recognize the user as a member. |
| **Actions:** | 1. The user asks to login. 2. The system asks the user to enter credentials. 3. The user enters credentials. 4. System forward credentials to Authenticator and Authenticator returns result to system. 5. If the entered credentials are correct, then the system recognizes the user as a member. 6. Otherwise, the system is showing incorrect credentials error message for the user and asking him to re-enter correct credentials or register to the market. |

|  |  |
| --- | --- |
| **Title:** | info about shop in the market and item in shop requesting (II.2.1) |
| **Actors:** | User. |
| **Preconditions:** | None |
| **Parameters:** | 1. Shop’s name. 2. Item name. |
| **Postconditions:** | None |
| **Expected result:** | The system must display information about the asked shop and item. |
| **Actions:** | 1. The user looks for a shop. 2. The system shows the shop’s information to the user. 3. The user looks for an item in the shop. 4. The system shows information about the item in this shop. |

|  |  |
| --- | --- |
| **Title:** | General items searching (II.2.2) |
| **Actors:** | User. |
| **Preconditions:** | None |
| **Parameters:** | At least one of the following:   1. Item’s name. 2. Item’s category. 3. Keywords. |
| **Postconditions:** | If the item does not exist, the system should show a relevant message. |
| **Expected result:** | The trading system displays the relevant items. |
| **Actions:** | 1. The user searches an item without specific shop. 2. The system displays the relevant items. 3. The user is able to filter the result by price range, item rank, category or shop rank. |

|  |  |
| --- | --- |
| **Title:** | Items saving in the shopping bag (II.2.3) |
| **Actors:** | User |
| **Preconditions:** | Items are added to the shopping bag. |
| **Parameters:** | Items. |
| **Postconditions:** | The system must remember the saved items in shopping bag for this user in some shop. |
| **Expected result:** | The items in the shopping bag of user in some shop is saved. |
| **Actions:** | 1. The user selects items to save in the shopping bag in some shop. 2. The system approves the request and remember the items in that shop for the requester user. |

|  |  |
| --- | --- |
| **Title:** | Shopping carts content checking and editing (II.2.4) |
| **Actors:** | User. |
| **Preconditions:** | None |
| **Parameters:** | Shopping cart. |
| **Postconditions:** | None |
| **Expected result:** | The system saves the changes were made by user in his shopping cart. |
| **Actions:** | 1. The user requests his shopping cart. 2. The system displays the shopping cart for the requester user, including items in the cart. 3. The user is able to see minimized info about his items, as well user can make changes on those items. |

|  |  |
| --- | --- |
| **Title:** | Shopping cart purchase for guest (II.2.5) |
| **Actors:** | Guest. |
| **Preconditions:** | None |
| **Parameters:** | Shopping cart. |
| **Postconditions:** | If the payment was done and the purchase was approved, then the system finishes the purchase.  Otherwise, the user gets a relevant message from the system on why purchase could not be done. |
| **Expected result:** | The purchase gets done for user and saved for history. |
| **Actions:** | 1. Guest asks to complete his purchase. 2. Shop executes following use cases sequency:    1. Use case: Purchase policy verifying (II.2.5.1).    2. Use case: Discount policy applying for guest (II.2.5.2.2)    3. The shop would ask visitor to continue to use case: Payment execution (II.3)    4. Use case: Purchase archiving (II.2.5.3) |

|  |  |
| --- | --- |
| **Title:** | Shopping cart purchase for Member (II.2.5 \*) |
| **Actors:** | Member. |
| **Preconditions:** | None |
| **Parameters:** | Shopping cart. |
| **Postconditions:** | If the payment was done and the purchase was approved, then the system finishes the purchase.  Otherwise, the user gets a relevant message from the system on why purchase could not be done. |
| **Expected result:** | The purchase gets done for user and saved for history. |
| **Actions:** | 1. Member asks to complete his purchase. 2. Shop executes following use cases sequency:    1. Use case: Purchase policy verifying (II.2.5.1).    2. Use case: Discount policy applying for member (II.2.5.2.1)    3. The shop would ask visitor to continue to use case: Payment execution (II.3)    4. Use case: Purchase archiving (II.2.5.3) |

|  |  |
| --- | --- |
| **Title:** | Purchase policy verifying (II.2.5.1) |
| **Actors:** | Shop |
| **Preconditions:** | Visitor asked to complete the purchase. |
| **Parameters:** | 1. Shopping basket 2. Visitor’s details |
| **Postconditions:** | If the shopping basket satisfies the purchase policy’s conditions the shop would approve.  Otherwise, error message would be shown to the visitor with relevant info and asks him/her to recheck his shopping basket. |
| **Expected result:** | Shop’s approval/rejection of visitor’s purchase according to purchase policy. |
| **Actions:** | 1. Visitors asks to complete a purchase. 2. The shop reviews the visitor’s shopping basket depending on its purchase policy. 3. If the purchase policy satisfies the purchase policy of the shop, the shop should approve the purchase.   Otherwise, shop should show error message with relevant broken conditions of the purchase policy. |

|  |  |
| --- | --- |
| **Title:** | Discount policy applying for member (II.2.5.2.1) |
| **Actors:** | Shop |
| **Preconditions:** | member asked to complete the purchase. |
| **Parameters:** | 1. Shopping basket 2. member’s details |
| **Postconditions:** | If discount policy specifies a discount for items were included in visitor’s shopping basket, prices must get updated accordingly.  Otherwise, prices should not be affected. |
| **Expected result:** | Shopping basket’s prices get updated according to discount policy of the shop. |
| **Actions:** | 1. Visitors asks to complete a purchase. 2. The shop reviews the items of the visitor’s shopping basket. 3. The shop checks for members’ discounts in the discount policy. 4. if there is a discount the shop would calculate the new price and update the shopping basket’s prices accordingly.   Otherwise, shop would execute use case (II.2.5.2.2). |

|  |  |
| --- | --- |
| **Title:** | Discount policy applying for guest (II.2.5.2.2) |
| **Actors:** | Shop |
| **Preconditions:** | Visitor asked to complete the purchase. |
| **Parameters:** | 1. Shopping basket 2. Visitor’s details |
| **Postconditions:** | If discount policy specifies a discount for items were included in visitor’s shopping basket, prices must get updated accordingly.  Otherwise, prices should not be affected. |
| **Expected result:** | Shopping basket’s prices get updated according to discount policy of the shop. |
| **Actions:** | 1. Visitors asks to complete a purchase. 2. The shop reviews the items of the visitor’s shopping basket. 3. The shop checks for guests’ discounts in the discount policy. 4. if there is a discount the shop would calculate the new price and update the shopping basket’s prices accordingly. |

|  |  |
| --- | --- |
| **Title:** | Purchase archiving (II.2.5.3) |
| **Actors:** | Shop |
| **Preconditions:** | Purchase was done successfully. |
| **Parameters:** | Shopping basket |
| **Postconditions:** | NONE |
| **Expected result:** | Purchase info is saved at purchase history of the shop. |
| **Actions:** | 1. Visitor completes his/her purchase successfully (after payment). 2. The shop asks purchase history to save the completed purchase. 3. Purchase history saves the purchase and buyer info. |

|  |  |
| --- | --- |
| **Title:** | Trading system quitting as a member (II.1.2\*) |
| **Actors:** | Member |
| **Preconditions:** | 1. Member is logged in |
| **Parameters:** | None |
| **Postconditions:** | 1. The member could find his shopping cart with its content in future visits. |
| **Expected result:** | 1. The system removes the member from the active members. 2. The system remembers the member’s shopping cart for future visits. |
| **Actions:** | 1. The user quits the trading system 2. The system removes the member from the active members. |

|  |  |
| --- | --- |
| **Title:** | Logout (II.3.1) |
| **Actors:** | Member |
| **Preconditions:** | The member is logged in. |
| **Parameters:** | None |
| **Postconditions:** | The shopping cart of the member is saved for future visits. |
| **Expected result:** | The user’s state is changed from member to guest. |
| **Actions:** | 1. The member requests to logout, 2. The system removes the member from active members, saves his shopping cart and change his state to guest. |

|  |  |
| --- | --- |
| **Title:** | Shop opining (II.3.2) |
| **Actors:** | Member |
| **Preconditions:** | Details of the new shop are unique. |
| **Parameters:** | Details of the new shop. |
| **Postconditions:** | 1. If the shop was opened the member should get success message, otherwise, system should send a relevant error message. 2. Other members can reach the new shop in the market. |
| **Expected result:** | The system opens a new shop with the specified details and assign the requester to be its first owner -founder-. |
| **Actions:** | 1. Member requests to open new shop in the trading system. 2. The system asks the member to enter shop’s details. 3. The system check that the provided details are unique and valid. 4. The system sends a message on success if so or error message if the process failed with information to the member. 5. The system assigns the requester member to be the founder of the shop as well, founder is an owner. |

# Shop owner/manager use cases

|  |  |
| --- | --- |
| **Title:** | Adding item to the shop’s stock (II.4.1.1) |
| **Actors:** | 1. Shop owner 2. Shop manager (with relevant permission) |
| **Preconditions:** | Item does not exist in the specific shop’s stock. |
| **Parameters:** | item details, shop details. |
| **Postconditions:** | Requester should get a response if the request either succeeded or failed. |
| **Expected result:** | The item is added to the specific shop’s stock. |
| **Actions:** | 1. Shop owner accesses a shop that is owned by him. 2. Shop owner requests to add new item to the stock of the specified shop. 3. The system asks the requester to enter the item’s details. 4. The requester enters the item’s details. 5. The system adds the item to the stock if does not exist, otherwise, system sends error messages to the requester. |

|  |  |
| --- | --- |
| **Title:** | Deleting item from shop’s stock (II.4.1.2) |
| **Actors:** | 1. Shop owner 2. Shop manager (with relevant permission) |
| **Preconditions:** | Item exists in the specific shop’s stock. |
| **Parameters:** | item details, shop details. |
| **Postconditions:** | Requester should get a response if the request either succeeded or failed. |
| **Expected result:** | The item is deleted from the specific shop’s stock. |
| **Actions:** | 1. Shop owner accesses a shop that is owned by him. 2. Shop owner requests to delete an item from the stock of the specified shop. 3. The system asks the requester to enter the item’s details. 4. The requester enters the item’s details. 5. The system searches and deletes the item from the stock the stock if exists, otherwise, system sends error messages to the requester. |

|  |  |
| --- | --- |
| **Title:** | Change item’s details in shop’s stock (II.4.1.3) |
| **Actors:** | 1. Shop owner 2. Shop manager (with relevant permission) |
| **Preconditions:** | Item exists in the specific shop’s stock. |
| **Parameters:** | item details, shop details. |
| **Postconditions:** | Requester should get a response if the request either succeeded or failed. |
| **Expected result:** | The changes of the item’s details are saved in the specific shop’s stock. |
| **Actions:** | 1. Shop owner accesses a shop that is owned by him. 2. Shop owner requests to change an item’s details from the stock of the specified shop. 3. The system asks the requester to enter the item’s details. 4. The requester enters the item’s details. 5. The system searches and changes the item’s details from the stock the stock if exists, otherwise, system sends error messages to the requester. |

|  |  |
| --- | --- |
| **Title:** | purchase policy changing (II.4.2.1) |
| **Actors:** | 1. Shop owner 2. Shop manager (with relevant permission) |
| **Preconditions:** | Shop exists. |
| **Parameters:** | Shop’s details, updated purchase policy’s details |
| **Postconditions:** | purchase policy is updated as requested if they do no conflicts with the consistency constraints were defined by the founder, otherwise error message is shown by the system to requester. |
| **Expected result:** | purchase policy is updated as requested. |
| **Actions:** | 1. Shop owner accesses his own shop and open the purchase policy. 2. The system displays the purchase policy with ability to change it. 3. Shop owner requests to update the purchase policy. 4. System asks shop owner to enter updated purchase policy’s details. 5. Shop owner enters the updated purchase policy’s details. 6. System verifies that details do not conflict with the consistency constraints were defined by the founder. 7. If all is valid, the system would approve those changes, otherwise, an error message is shown to the requester by the system. |

|  |  |
| --- | --- |
| **Title:** | Discount policy changing (II.4.2.2) |
| **Actors:** | 1. Shop owner 2. Shop manager (with relevant permission) |
| **Preconditions:** | Shop and items are existing. |
| **Parameters:** | Shop’s details, items’ details and their updated discount policy. |
| **Postconditions:** | Discount policy is updated as requested if they do no conflicts with the consistency constraints were defined by the founder, otherwise error message is shown by the system to requester. |
| **Expected result:** | Discount policy is updated as requested. |
| **Actions:** | 1. Shop owner accesses his own shop and open the discount policy. 2. The system displays the discount policy with ability to change it. 3. Shop owner requests to update the discount policy. 4. System asks shop owner to enter updated discount policy’s details. 5. Shop owner enters the updated discount policy’s details for. 6. System verifies that details do not conflict with the consistency constraints were defined by the founder. 7. If all is valid, the system would approve those changes, otherwise, an error message is shown to the requester by the system. |

|  |  |
| --- | --- |
| **Title:** | Shop owner assignment (II.4.4) |
| **Actors:** | 1. Shop owner 2. Shop manager (with relevant permission) |
| **Preconditions:** | Assigned member exists and he is not an owner of the specific shop. |
| **Parameters:** | Member’s details and shop’s details. |
| **Postconditions:** | If the assignment was done successfully the system saves the assigner of the new owner.  Otherwise, the system shows a relevant error message. |
| **Expected result:** | The specified member is defined to be an owner of the specified shop and sends an approval. |
| **Actions:** | 1. Shop owner accesses a shop that is owned by him. 2. The system displays the shop with ability to assign a new owner. 3. The shop owner requests to add a new owner to his shop. 4. The system asks the requester to enter member’s details to assign him as an owner. 5. The shop owner enters the member’s details. 6. The system checks if this member is not already an owner of the specified shop, then system defines the member as an owner, otherwise the system shows a relevant error message. |

|  |  |
| --- | --- |
| **Title:** | Shop manager assignment (II.4.6) |
| **Actors:** | 1. Shop owner 2. Shop manager (with relevant permission) |
| **Preconditions:** | The specified member is not a manger or an owner of the specified shop. |
| **Parameters:** | Shop’s details and member’s details. |
| **Postconditions:** | Two following default permissions are granted to new manager:   1. Use case 4.12 2. Use case 4.13   If the assignment was done successfully the system saves the assigner of the new manager.  Otherwise, the system shows a relevant error message. |
| **Expected result:** | The system defines the specified member as a manger of the specified shop and sends an approval. |
| **Actions:** | 1. Shop owner accesses a shop that is owned by him. 2. The system displays the shop with ability to assign a new manager. 3. The shop owner requests to add a new manager to his shop. 4. The system asks the requester to enter member’s details to assign him as a manager. 5. The shop owner enters the member’s details. 6. The system checks if this member is not already a manager or an owner of the specified shop, then system defines the member as a manager, otherwise the system shows a relevant error message. 7. System adds 4.12 and 4.13 default permissions to the new manager of the shop. |

|  |  |
| --- | --- |
| **Title:** | Shop manager permissions updating (II.4.7) |
| **Actors:** | 1. Shop owner 2. Shop manager (with relevant permission) |
| **Preconditions:** | The specified manager was assigned as a manger by this owner. |
| **Parameters:** | Shop’s details, manager’s details and updated permission. |
| **Postconditions:** | If permission changing succeeded the system returns an approval message, otherwise a relevant error message is shown by the system. |
| **Expected result:** | Permissions of the specified are updated due to the given updated permission. |
| **Actions:** | 1. Shop owner accesses a shop that is owned by him. 2. The system displays the shop with ability to access roles with an ability to update manager’s permissions. 3. The shop owner requests to update the specified manager’s permissions with the specified updated permissions. 4. The system asks the requester to enter manager’s details and updated permission’s details to update his permissions. 5. The shop owner enters the member’s details and updated permission’s details. 6. The system updates the permission of the specified manager due to given specified permissions, otherwise the system shows a relevant error message. 7. System sends an approval of manager’s permissions updating. |

|  |  |
| --- | --- |
| **Title:** | Shop closing (II.4.9) |
| **Actors:** | 1. Shop owner 2. Shop manager (with relevant permission) |
| **Preconditions:** | Shop owner owns the given shop. |
| **Parameters:** | Shop’s details. |
| **Postconditions:** | 1. The closed shop and its items are not visible for member who are not owner and manager of it anymore via search results in the market. 2. Owners and managers get notified about the shop closing. 3. The system keeps the roles saved. |
| **Expected result:** | The specified shop is closed. |
| **Actions:** | 1. Shop owner accesses a shop that is owned by him. 2. The system displays the shop with ability to close it. 3. The shop owner requests to close the given shop. 4. system saves the roles of the shop and define it as an inactive shop. 5. System does not show the shop and its items in search results anymore. 6. System notifies the owners and managers of the system. |

|  |  |
| --- | --- |
| **Title:** | Shop’s roles info request (II.4.11) |
| **Actors:** | 1. Shop owner 2. Shop manager (with relevant permission) |
| **Preconditions:** | The shop exists, the shop owner owns the specified shop. |
| **Parameters:** | Shop’s details and member’s details. |
| **Postconditions:** | If the specified member is a role holder the system will provide details about him, as well if he is a shop manager, his permission would be shown.  Otherwise, shop owner should get an error message. |
| **Expected result:** | The system will provide details about him, as well if he is a shop manager, his permission would be shown. |
| **Actions:** | 1. Shop owner accesses a shop that is owned by him. 2. System shows the shop owner details about the specified shop with ability to request information about roles holders in his shop. 3. The shop owner request information about role holder. 4. System asks requester to enter role holder’s details. 5. The requesters enter the details. 6. System checks if details are valid and relevant to a role holder, if this member is a manager system will show information about his permission in addition to details about the role, otherwise if he is not a role holder the system shows an error message. |

|  |  |
| --- | --- |
| **Title:** | In-shop purchases history request (II.4.13)/( (II.6.4) |
| **Actors:** | 1. Market manager 2. shop owner 3. Shop manager (with relevant permission) |
| **Preconditions:** | 1. The shop exists in the market. 2. If actor is shop owner/manager, then specified shop must be owned/managed by him. |
| **Parameters:** | Shop’s details. |
| **Postconditions:** | Report is returned in case of success; otherwise relevant error message is shown. |
| **Expected result:** | The market manager/shop owner/manager gets a report of in-shop purchase history. |
| **Actions:** | 1. Actor requests from system the in-shop purchase history. 2. System checks permissions. 3. System returns a report summarize the in-shop purchases. |