Use case: **Register to system**

* + **Actor: Guest User**
  + **Precondition:** Guest is not already registered to the system
  + **Parameter:** Member information
  + **Actions:**
    1. **Guest** enters the system
    2. **System** presents the option to signup
    3. The **Guest** provides the **Member** **information**
    4. **System** returns indication that the user is now a **member**

|  |  |
| --- | --- |
| Action | Expected Result |
| Provide valid member information that isn't currently in the system. | A new member has been added to the system withthe provided member information. |
| Provide member information that is currently in the system. | Error message |
| Provide invalid member information. | Error message |
|  |  |

* Use case: **Login**
  + **Actor: Guest User**
  + **Precondition:** User is a member of the system.
  + **Parameter:** Email and password
  + **Actions:**
    1. **Guest** enters the system
    2. **System** presents the option to enter email address and password and to login
    3. **Guest User** enters his email address and password
    4. If provided **Guest User**'s information is in the system and his password is valid
       1. **System** identifies the **Guest User** as member
       2. **System** informs that the member successfully logged in

|  |  |
| --- | --- |
| Action | Expected Result |
| Provide valid login information that is currently in the system. | The user is being logged in the system. |
| Provide login information that isn't currently in the system. | Error message |
| Provide invalid login information. | Error message |

* Use case: **Gather information about store/product**
  + **Actor: User**
  + **Precondition:** User opened the system
  + **Parameter:** String (presenting a product or a store)
  + **Actions:**
    1. **User** provides the product search information of the **product/store** he wants information of
    2. **System** searches through all it's system database for:
       1. **Products** that its name matches the provided information
       2. **Stores** that its name matches the provided information.
    3. **System** presents all the matching stores and products.

|  |  |
| --- | --- |
| Action | Expected Result |
| User Provides information about product that is currently in the system. | The system shows the relevant products matching to the provided information |
| Provide information that isn't currently in the system. | Message that indicates there are no matching products. |
| User Provides information about a store that is currently in the system. | The system shows the relevant stores matching to the provided information. |

* Use case: **Search for products**
  + **Actor: Buyer**
  + **Precondition:** There are products in the system
  + **Parameter:** search term
  + **Actions:**
    1. **Buyer** types in a string representing what he looks for
    2. **System** searches through all the **products** in the data base- by its name and description
    3. **System** presents all the best matching results in a list
    4. **Buyer** (optional) chooses Parameters to filter the products listed (such as price, category and so on)
    5. **System** filters the products and presents the new results.

|  |  |
| --- | --- |
| Action | Expected Result |
| User enters valid string of product name to search in the system, and such product exists in the system | The system shows the relevant products matching to the provided string |
| User enters valid string of product name to search in the system, and no such product exists in the system | Message that indicates there are no matching products. |
| User enters invalid string | **Error message** |

* Use case: **Add product to basket**
  + **Actor: Buyer**
  + **Precondition:** There are available products to buy
  + **Parameter:** Products
  + **Actions:**
    1. System allows user to use Search for product use case
    2. After searching, buyer chooses a product to add to basket, and amount
    3. System updates the current store's basket
       1. System checks if the product already exists in the current store's basket
          1. If so- updates the amount
          2. Else- adds new product to the basket with the requested amount

|  |  |
| --- | --- |
| Action | Expected Result |
| User chooses product to buy | The system adds this product to the user's shopping cart. |
| User chooses product to buy and the owner just removed the product | The system shows message indicates that the product is no longer available |
| User enters a negative amount of the product. | The system shows message indicates that the amount is no valid. |

* Use case: **View shopping cart**
  + **Actor: Buyer**
  + **Precondition:**
  + **Parameter:**
  + **Actions:**
    1. **Buyer** asks to present the shopping cart
    2. **System** presets all the products in the shopping cart, their amount and total amount and price in the cart

|  |  |
| --- | --- |
| Action | Expected Result |
| User request to present shopping cart- when the shopping cart isn't empty. | The system presents the requested shopping cart. |
| User request to present shopping cart- when the shopping cart is empty. | The system presents the requested shopping cart. |
| User request to present shopping cart- when the shopping cart is empty. | The system shows an empty list of products and the total price will be 0. |

* Use case: **Edit shopping cart-**
  + **Actor: Buyer**
  + **Precondition:** Shopping cart not empty
  + **Parameter:** Product to update and chosen change to make
  + **Actions:**
    1. **System** presents the existing **shopping cart**
    2. **Buyer** choose a **product** from the **cart**
    3. If the **Buyer** choose changing the amount
       1. If the new requested amount is 0, the **product** is removed from the **cart**
       2. Else the amount is updated as requested

|  |  |
| --- | --- |
| Action | Expected Result |
| User adds a valid amount of a product (i.e less then what the Store is offering) | The system adds the required products and presents the requested shopping cart. |
| User reduces the amount of a product to a number above 0 | The system the reduces the amount in the cart as requested |
| User reduces the amount of a product to 0 | The system removes the product from the user’s cart |
| User reduces the amount of a product to a negative number | The system removes the product from the user’s cart |

* **Use case: buy all products in cart**
  + Actor: User
  + Precondition: Shopping cart not empty
  + Parameter: Cart- list of baskets to buy
  + Actions:
    1. User asks to buy all products in the cart
    2. System runs basket buying for each basket
       1. System checks validity of products in the cart-
          1. System checks for basic logics- amounts
          2. System send request to store's policy check with products
          3. Store validates products with store's policy
       2. System checks for discounts with store's discount policy
       3. Store's discount policy calculates and returns updated prices for products
       4. System checks with store for availability of all products, that all are in stock
       5. Store responds that all products are in stock- available
    3. System updates all information and presenting to user
    4. User accepts and move to buying process
    5. System asks for payment information
    6. User provides payment information requested
    7. System sends payment information to payment proxy- to connect to external payment system
    8. External payment system responds with positive feedback
    9. System sends shipping request to a shipping proxy- to connect to an external shipping system
    10. External shipping system responds with positive feedback
    11. System informs the user that all the products were successfully purchased and are shipped to the user
* Use case: **Logout**
  + **Actor: Member**
  + **Precondition: Member** is logged in to the **system**
  + **Parameter:**
  + **Actions:**
    1. **Member** asks to logout from the **system**
    2. **System** changes **user** status from logged in **~~member~~** to **guest**
    3. **Member** can keep use the **system** as a guest(not logged in member)

|  |  |
| --- | --- |
| Action | Expected Result |
| The user logging out from the system | The user status changed to guest status, and he can use the system as a guest |

* Use case: **Open a store**
  + **Actor: Member**
  + **Precondition: Member** is logged in to the **system**
  + **Parameter:** New **store** information
  + **Actions:**
    1. **Member** asks to open a new store in the **system**
    2. **System** asks for information about the **store** from the **user**
    3. **Member** provides information
    4. **System** adds the store to the **system**
       1. **System** signs the **Member** as the founder of the **store**
       2. \*\*\*Should go to edit store policy

|  |  |
| --- | --- |
| Action | Expected Result |
| The user is logged to the system and provides valid information while opening the store | The store is added to the system with the provided information and the user is signed as co-founder of the store |
| The user is logged to the system and provides non valid information while opening the store | Message indicates the information provided for creating a store is invalid showed to the user. |
| The user is not logged to the system and provides valid information while opening the store | Message indicates that creating a store required the user to be logged in. |

* Use case: **Review purchase history**
  + **Actor: Member**
  + **Precondition: Member**is logged in to the **system**
  + **Parameter:**
  + **Actions:**
    1. **System** presents the option to view personal purchase history
    2. **Member** asks to view the history
    3. System presents details about every previous the user performed in the past

|  |  |
| --- | --- |
| Action | Expected Result |
| The user is logged to the system and asks to watch his personal purchase history | A list of all the user's previous purchases. |

* Use case: **Add new product to store**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
    3. **Same product does not already exist in the store**
  + **Parameter:** Store identification and new product
  + **Actions:**
    1. Member asks to add a new product to a store he owns
    2. The system asks for store identification
    3. Member enters store id
    4. System asks for product's details
    5. **Member** provides required details- product details
    6. System checks if such a program already exist in the system
    7. If the product with the same product id exists already in the stor
       1. System informs Member that a product already exists
    8. Else
       1. System adds the product to the store's products resource
       2. System updates product's current stock to 0
       3. System informs the Member that the product was added successfully

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns and valid new product details | * The system added the new product to the store * The product can be found when searching for products in the store * The product can be found when searching for products in all the system * The product's stock is 0 |
| Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store he owns, and of a product that already exists in the store | System informs the **Member that the products he asks to add already exits in the store** |
| The Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |
| The Member is logged to the system and provides identification of a store he owns, and of a product that doesn't exists, and with invalid product details | System informs the **Member** that the product's details are invalid |

* Use case: **Remove product from store**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** Store identification and product id
  + **Actions:**
    1. Member asks to remove a product from a store he owns
    2. The system asks for store identification
    3. Member enters store id
    4. System asks for product's id
    5. **Member** provides required details- product id
    6. System checks if such a program already exist in the system
    7. If the product with the same product id exists in the store
       1. System deletes product from the store resources
       2. System informs Member that a product was deleted successfully from the store
    8. Else
       1. System informs the **Member** that the product doesn’t exists in the store

|  |  |
| --- | --- |
| Action | Expected Result |
| Action: Member is logged to the system and provides an identification of an existing store that he owns and an a product id that exists in the store | * The system deleted the product from the store * The product can't be found when searching for products in the store |
| Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store he owns, and of a product that doesn't exists in the store | System informs the **Member that the products he asks to add doesn't exists in the store** |
| The Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |

* Use case: **Update product stock- add items**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** Store identification, product id, amount
  + **Actions:**
    1. Member asks to add product items to store's stock
    2. The system asks for store identification, product id and amount
    3. Member enters store id
    4. **Member** provides required details- store id, product id, amount
    5. System checks if such a store exists in user's store repository
    6. If exists
       1. System checks if such product exists in the store
       2. If exists
          1. System updates the product's current stock to the new amount
       3. Else
          1. Store asks if the **Member** wants to add the new product to the store
          2. If **Member** choose to add the product

**Member** start **Add new product to store** use case

System updates product's stock to the new amount

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns and an a product id that exists in the store, and a valid amount | The system updated the product's stock in the store |
| Action: Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store he owns, and of a product that doesn't exists in the store | System informs the **Member** that the products he asks to add doesn't exists in the store |
| The Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |
| The Member is logged to the system and provides a store id of an existing store he owns and a product id that exits in that store, and an invalid amount | Message indicates that requested amount is invalid |

* Use case: **Update product stock- subtract items**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** Store identification, product id, amount
  + **Actions:**
    1. Member asks to subtract product items from store's stock
    2. The system asks for store identification, product id and amount
    3. **Member** provides required details- store id, product id, amount
    4. System checks if such a store exists in user's store repository
    5. If exists
       1. System checks if such product exists in the store
       2. If exists
          1. System updates the product's current stock to the new amount
       3. Else
          1. System informs the **Member** that such product doesn't exist in the system

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns and an a product id that exists in the store, and a valid amount to subtract | The system updated the product's stock in the store |
| Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store he owns, and of a product that doesn't exists in the store | System informs the **Member** that the products he asks to add doesn't exists in the store |
| The Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |
| The Member is logged to the system and provides a store id of an existing store he owns and a product id that exits in that store, and an invalid amount | Message indicates that requested amount is invalid |

* Use case: **Update existing product's details**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** Store identification, product id, product details
  + **Actions:**
    1. Member asks to update product's details
    2. The system asks for store identification, product id and new details
    3. **Member** provides required details- store id, product id, new details
    4. System checks if such a store exists in user's store repository
    5. If exists
       1. System checks if such product exists in the store
       2. If exists
          1. System updates the product's details as required

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns and an a product id that exists in the store, and a valid new details | The system updated the product's details |
| Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store he owns, and of a product that doesn't exists in the store | System informs the **Member** that the products he asks to add doesn't exists in the store |
| The Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |
| The Member is logged to the system and provides a store id of an existing store he owns and a product id that exits in that store, and invalid new details | Message indicates that requested details are invalid |

* Use case: **Add buying strategy to store's policy**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** Store identification, product id, product details
  + **Actions:**
    1. Member asks to update product's details
    2. The system asks for store identification, product id and new details
    3. **Member** provides required details- store id, product id, new details
    4. System checks if such a store exists in user's store repository
    5. If exists
       1. System checks if such product exists in the store
       2. If exists
          1. System updates the product's details as required

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns and an a product id that exists in the store, and a valid new details | The system updated the product's details |
| Member is logged to the system and provides identification of a store that doesn't exist | **:** **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| : The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store he owns, and of a product that doesn't exists in the store | System informs the **Member** that the products he asks to add doesn't exists in the store |
| : The Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |
| The Member is logged to the system and provides a store id of an existing store he owns and a product id that exits in that store, and invalid new details | Message indicates that requested details are invalid |

* Use case: **Add buying strategy to store's policy**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** store id, product id and buying strategy
  + **Actions:**
    1. **Member asks to add buying strategy to product**
    2. **System requests the store id, buying strategy and product id**
    3. **Member provides required information**
    4. **If store exists, buying strategy valid and product exists in store**
       1. **System check if requested buying strategy exists in store's policy**
       2. **If exits in policy - System adds buying strategy to product**
    5. **Else**
       1. **System informs user**

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns and an a new buying strategy to store's policy | The system updated the store's policy |
| Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| Member is logged to the system and provides identification of a store he owns and an unknown buying strategy | System informs the **Member** that the buying strategy is unknown |
| Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |

* Use case: **Update buying strategy to store's policy**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** store id and buying strategy
  + **Actions:**
    1. **Member asks to update buying strategy to store**
    2. **System requests the store id and strategy**
    3. **Member provides required information**
    4. **System locates store and check if strategy exists in the store's policy**

**Else:**

**Message indicates the strategy doesn't exist will be shown to the user**

* + 1. **System asks whether to delete or update existing strategy**
       1. **If member chooses to update**
          1. **System update the strategy in the store with the new strategy provided**
       2. **If member chooses to delete**
          1. **System removes strategy from store's policy**

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns and an existing buying strategy in store's policy | The system updated the store's policy as required |
| Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store he owns and an unknown buying strategy | System informs the **Member** that the buying strategy is unknown |
| The Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |
| The Member is logged to the system and provides identification of a store he owns and a strategy that doesn't exist | System informs the **Member** that the buying strategy doesn't exist and encourages him to use **Add new strategy to store's policy** use case |

* Use case: **Add allowed discounts to store's policy**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** store id and discount type
  + **Actions:**
    1. **User asks to add allowed discount type to store**
    2. **System requests the store id and discount type**
    3. **User provides required information**
    4. **If strategy discount type exists**
       1. **System doesn't change it**
    5. **Else**
       1. **System adds new discount type to store's policy**

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns and an a new discount type to store's policy | The system updated the store's policy |
| Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store he owns and an unknown discount type | System informs the **Member** that the **discount type** is unknown |
| The Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |

* Use case: **Update allowed discounts in store's policy**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** store id and discount type
  + **Actions:**
    1. **User asks to update** discount type **to store.**
    2. **System requests the store id and** discount type
    3. **User provides required information**
    4. **System locates store and check if** discount type **exists in the store's policy**

**If not: Relevant message will be shown to the member.**

**If so:**

* + 1. **System asks whether to delete or update existing** discount type
       1. **If user chooses to update**
          1. **System update the** discount type **in the store's policy with the new** discount type **provided**
       2. **If user chooses to delete**
          1. **System removes** discount type **from store's policy**

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns and an existing discount type in store's policy | The system updated the store's policy as required |
| Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| : The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store he owns and an unknown discount type | System informs the **Member** that the discount type is unknown |
| The Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |
| The Member is logged to the system and provides identification of a store he owns and a discount type that doesn't exist | System informs the **Member** that the discount type doesn't exist and encourages him to use **Add new discount type to store's policy** use case |

* Use case: **View store's policy**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** store id and discount type
  + **Actions:**
    1. **User asks to view store's policy**
    2. **System asks for store's id**
    3. **User provides store id**
    4. **If all the provided information match specific store:**

**System presents store policy**

**Else:**

**Relevant message will be shown to the member.**

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns | The system presents store's policy- buying strategies and discount types |
| Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is not logged to the system | Message indicates that managing store's policy is only allowed to logged in existing members |

* Use case: **View Store employees and permissions**
* **Actor: Member**
* **Preconditions:** Member is logged into the System and he is the owner of the provided store
* **Parameter:** store id
* **Actions:**

1. Member asks to view the store employees and permissions
2. System requests a store id
3. Member provides the store id
4. **If all the provided information match specific store:**

System presents all employees in the store and their permissions

**Else:**

**Relevant message will be shown to the member.**

**Acceptance tests:**

|  |  |
| --- | --- |
| Action | Expected result |
| Member provides a valid Store id of which he is an owner | **System presents the requested information** |
| Member provides a store id that is not in the system | **Error message** |
| Member provides a store id of a store of which he is not an owner | **Error message** |

* **Use case: Nominate member to be store manager**
  + **Actor: Member**
  + **Precondition:** 
    1. **Member** is logged in to the **system**
    2. **Member** is an owner of an existing store
  + **Parameter:** store id, user id, permissions
  + **Actions:**
    1. **Member asks to nominate an existing member to be manager of the store**
    2. **System requests the store id and user id**
    3. **User provides required information**
    4. **System locates store and user**
    5. **System asks for permission to add to user**
    6. **User provides the permissions**
    7. **If all valid**
       1. **System assigns requested user to be store manager**
       2. **System adds permissions to user**
       3. **System sets user permission as new manager**
       4. **System sets user to be nominated by the user that asked for this nomination**
       5. **Requested user informs that he got the new nomination and by who**

**Else:**

**Relevant message will be shown to the member.**

|  |  |
| --- | --- |
| Action | Expected Result |
| Member is logged to the system and provides an identification of an existing store that he owns, and id of a user of the system, that is not already a store manager | The system adds the user as manager of the store, updates his permissions and set his nominator to be the user that nominated him |
| Member is logged to the system and provides identification of a store that doesn't exist | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store that he doesn’t own | **System** informs **Member** that the **store** doesn’t exist in his owned store repository |
| The Member is logged to the system and provides identification of a store he owns and an unknown user | System informs the **Member** that the **user is unknown to the system** |
| The Member is not logged to the system | Message indicates that managing stock is only allowed to logged in existing members |
| The Member is logged to the system and provides identification of a store he owns and a known user that is already a manager of that store | Message indicates that user can't be nominated twice to be store manager of the same store |
| The Member is logged to the system and provides identification of a store he owns and doesn't have permissions to nominate new manager | Message indicates that user can't nominate new manager to store because of permissions issues |

* Use case: **Change management permission for sub-manger**
  + **Actor: Member**
  + **Precondition:** The Member is owner of the store the sub mangers set is manages and the set is not empty.
  + **Parameter:** Set of permissions
  + **Actions:**
    1. **Member** chooses sub-manger from an non empty sub mangers set for specific store.
    2. The **Member** sets a set of permissions for the selected sub manager.
    3. If The set of permissions is valid:

The sub manager gets the new permissions.

Else

The sub manager stays with the former permissions.

|  |  |
| --- | --- |
| Action | Expected Result |
| The member is owner of the store the sub mangers set is manages and the sub mangers set is not empty the Store's owner selects sub manager and give him valid set of permissions, the sub manager does operations required each of the permissions from the set of permissions. | All the operations succeeded. |
| The member is owner of the store the sub mangers set is manages and the sub mangers set is not empty the Store's owner selects sub manager and give him valid set of permissions, the sub manager does operation that required permission that doesn't belongs to the provided set of permissions | Message indicates the sub manger doesn’t have the permissions shown to the sub manager. |
| The member is owner of the store the sub mangers set is manages and the sub mangers set is not empty the Store's owner selects sub manager and give him non valid set of permissions, the sub manager does operations required each of the permissions from the set of permissions. | Message indicates the set of permissions is not valid will be shown the Store's Owner. |

* Use case: **Member Remove management permission for sub-manger**
  + **Actor: Member**
  + **Precondition:** The member owns the given store and sub manager is actual sub manager of the store.
  + **Parameter:** Store andsub manager.
  + **Actions:**
    1. The Store's Owner removes the sub manager from the managers set of the store

**2.**The right management permissions removed from the former sub manager permissions for the specific store

|  |  |
| --- | --- |
| Action | Expected Result |
| The member owns the store and the Store's Owner removes sub manager's permission for specific store, the sub manager tries to do operation that requires management permission for the specific store | The operation aborted and the sub manager get appropriate message. |
| The member owns the store and the Store's Owner removes management permission for sub manager which is not sub manager of the specific store | Message indicates that the sub manager is not manager of the specific store will be shown the Store's Owner. |
| The member owns the store and the Store's Owner removes sub manager's permission for specific store, the sub manager tries to do operation that not requires management permission for the specific store | the operation succeeded and former sub manager get approval message for the operation. |

* Use case: **Member requests for purchase history for the store**
  + **Actor: Member**
  + **Precondition:** The member owns the store.
  + **Parameter:** Store
  + **Actions:**
    1. The member requests to print the history of purchase for specific range of dates
    2. If The member has ownership permissions for the specific store:

2.1 The user get the requested information

**2.**Else:

2.1: The user get a message for inappropriate permissions.

|  |  |
| --- | --- |
| Action | Expected Result |
| The member owns the store and the Store's Owner management permission for the specific store, the owner changes price of a product that belongs to the purchase history | The history printed successfully with the right price of the changed product |
| The member doesn't owns the store and the Store's Owner management permission for the specific store, the user changes price of a product that belongs to the purchase history | Message indicates that user is not owner of the specific store will be shown the Store's Owner |
| The member owns the store and the Store's Owner management permission for the specific store, the owner changes details of a product that belongs to the purchase history | The history printed successfully with the right price of the changed product |

* Use case: **Admin requests for user history** 
  + **Actor: Admin**
  + **Precondition:** user belongs to the system
  + **Parameter:** user identifier
  + **Actions:**
    1. The user requests to print the history of provided user identifier for specific range of dates
    2. If The user has ecommerce management permissions:

2.1 The user get the requested information

**2.**Else:

2.1: The user get a message for inappropriate permissions.

|  |  |
| --- | --- |
| Action | Expected Result |
| The provided user identifier belongs to the system and user requests to print the history of provided user identifier for specific range of dates | The history printed successfully. |
| The user is not manager of the ecommerce system the provided user identifier belongs to the system and the user requests to print the history of provided user identifier for specific range of dates | Message indicates that user is not manager of the ecommerce system be shown the user. |
| The user is manager of the ecommerce system the provided user identifier doesn't belongs to the system and the user requests to print the history of provided user identifier for specific range of dates | Message indicates that provided user is not belongs to the ecommerce system be shown the user. |