E-commerce: Use Cases

1. User Manage: 3
- [1.1] Use case: Connect to system 3
- [1.2] Use case: Disconnect to system....................3
- [1.3] Use case: Register to system 3
- [1.4] Use case: Login 4
- [1.5] Use case: Logout 4
- [1.6] Use case: Review purchase history....................5
- [1.7] Use case: Appoint user to be store co-owner 5
- [1.8] Use case: Appoint Manager....................6
- [1.9] Use case: Update management permission for sub-manger 7
- [1.10] Use case: Remove management permission for sub-manger....................8
- [1.11] Use case: Review store's stakeholders 9
- [1.12] Use case: Admin requests for all users history....................9
- [1.13] Use case: Admin requests for store history....................10
- [1.14] Use case: Remove Co-owner from Store....................10

2. Items and stores:....................12
- [2.1] Use case: Gather information about store/product....................12
- [2.2] Use case: Search for products 12

3. User buying from stores: 13
- [3.1] Use case: Save items in a shopping cart 13
- [3.2] Use case: View shopping cart 13
- [3.3] Use case: Edit shopping cart-....................14
- [3.4] Use case: Purchase the whole cart 14
- [3.5] Use case: Calculate total price for basket....................16
- [3.6] Use case: Offer Purchase for a Product 17

4. Store Manage: 17
- [4.1] Use case: Open a store 17
- [4.2] Use case: Add new product to store....................18
- [4.3] Use case: Remove product from store 19
- [4.4] Use case: Update product stock- add items 20
- [4.5] Use case: Update product stock- subtract items....................21

- [4.6] Use case: Update existing product's details 22
- [4.7] Use case: Add buying strategy to store's policy 23
- [4.8] Use case: Update buying strategy to store's policy 24
- [4.9] Use case: Add allowed discounts to store's policy 25
- [4.10] Use case: Update allowed discounts in store's policy 26
- [4.11] Use case: View store's policy 27
- [4.12] Use case: Add buying strategy to store's product 27
- [4.13] Use case: Update buying strategy to store's product... 28
- [4.14] Use case: View product's buying strategies 29
- [4.15] Use case: Add discount to product 30
- [4.19] Use case: Store Owner get purchase history of a store 31

1. User Manage:

[1.1] Use case: Connect to system

- **[Req: 2.1, Class: User Manager, CNAME: Connect]**
- **Actor: Guest User**
- **Precondition:** System is disconnected
- **Parameter:**
- **Actions:**
  1. **System** presents the option to connect
  2. **Guest** chooses to connect
  3. **System** connects the user to use the system's options

Action	Expected Result
User chooses to connect	System connects user to the system

[1.2] Use case: Disconnect to system

- **[Req: 2.2, Class: User Manager, CNAME: Disconnect]**
- **Actor: Guest User**
- **Precondition:** Guest is connected to the user
- **Parameter:**
- **Actions:**
  1. **System** presents the option to disconnect
  2. **Guest** chooses to disconnect from the system
  3. **System** disconnects user from the system, and the user can perform actions in the system.

Action	Expected Result
User chooses to disconnect	System disconnects the user
User is in the middle of a process- buying the cart, and asks to disconnect	System disconnects the user and keeps performs the action in the background

[1.3] Use case: Register to system

- **[Req: 2.3, Class: User Manager, CNAME: Register, Test:TestRegister]**
- **Actor: Guest User**
- **Precondition:** Guest is not already registered to the system
- **Parameter:** Member information
- **Actions:**
  1. **System** presents the option to signup
  2. The **User** provides the **Member information**
  3. **User** enters his chosen password
  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 with the provided member information.
Provide member information that is currently in the system.	Error message
Provide invalid member information.	Error message

[1.4] Use case: Login

- **[Req: 2.4, Class: User manager, CNAME: Login, Test:TestLogin]**
- **Actor:** **Guest User**
- **Precondition:** User is a member of the system.
- **Parameter:** Username and password
- **Actions:**
  1. **Guest** enters the system
  2. **System** presents the option to enter username and password to login
  3. **Guest User** enters his username and password
  4. If provided **Guest User**'s information is in the system and his password is valid
     a) **System** identifies the **Guest User** as member
     b) **System** informs that the member successfully logged in
  5. Else
     a) System informs the user that the provided information is incorrect

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

[1.5] Use case: Logout

- **[Req: 3.1, Class: User Manager, CNAME: Logout, Test:TestLogout]**
- **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 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

[1.6] Use case: Review purchase history

- **[Req: 3.7, Class: User, CNAME: PersonalPurchaseHisotry, Test:TestViewPurchaseHistory]**
- **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 purchase history information about every previous purchase 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.

[1.7] Use case: Appoint user to be store co-owner

- **[Req: 4.3, Class: User, CNAME:AppointCoOwner, Test:TestAppointCoOwner]**
- **Actor: Store owner**
- **Precondition:**
  1. **Store owner** is logged in to the **system**
  2. **Store owner** is an owner of an existing store
- **Parameter:** store id, appointee user id
- **Actions:**
  1. User asks to appoint an existing user to be manager of the store
  2. System requests the store id and appointee user id
  3. User provides required information
  4. System locates store and users
  5. If all valid
     a) System assigns requested user to be store owner
     b) System adds permissions to user
     c) System sets user permission as new owner
     d) System sets appointee user to be appointed by the user that asked for this nomination
     e) Requested user informs that he got the new nomination and by who

Action	Expected Result
Store owner 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 owner	The system adds the user as owner of the store, updates his permissions and set his nominator to be the user that nominated him

Store owner is logged to the system and provides identification of a store that doesn't exist	**System** informs **Store owner** that the **store** doesn’t exist in his owned store repository
The Store owner **is logged to the system and provides identification of a store that he doesn’t own**	**System** informs **Store owner** that the **store** doesn’t exist in his owned store repository
The Store owner **is logged to the system and provides identification of a store he owns and an unknown** user	System informs the **Store owner** that the **user is unknown to the system**
The Store owner is not logged to the system	Message indicates that managing stock is only allowed to logged in existing members
The Store owner is logged to the system and provides identification of a store he owns and a known user that is already a owner of that store	Message indicates that user can't be nominated twice to be store owner of the same store
The Store owner is logged to the system and provides identification of a store he owns and doesn't have permissions to nominate new owner	Message indicates that user can't nominate new owner to store because of permissions issues

[1.8] Use case: Appoint Manager

- **[Req: 4.5, Class: User, CNAME: AppointManager, Test:TestAppointManager]**
- **Actor: Store owner**
- **Precondition:**
  1. **Store owner** is logged in to the **system**
  2. **Store owner** is an owner of an existing store
- **Parameter:** store id, user id
- **Actions:**
  1. User asks to appoint an existing user to be manager of the store
  2. System requests the store id and appointee user id
  3. User provides required information
  4. System locates store and user
  5. If all valid
     a) System assigns requested user to be store manager
     b) System adds basic permissions to appointee user
     c) System sets user permission as new manager
     d) System sets user to be nominated by the user that asked for this nomination
     e) Requested user informs that he got the new nomination and by who

Action	Expected Result
Store owner is logged to the system and provides an identification of an existing store	The system adds the user as manager of the store, updates his permissions and set

that he owns, and id of a user of the system, that is not already a store manager	his nominator to be the user that nominated him
Store owner is logged to the system and provides identification of a store that doesn't exist	**System** informs **Store** that the **store** doesn't exist in his owned store repository
The Store owner is logged to the system and provides identification of a store that he doesn't own	**System** informs **Store owner** that the **store** doesn't exist in his owned store repository
The Store owner is logged to the system and provides identification of a store he owns and an unknown user	System informs the **Store owner** that the **user is unknown to the system**
The Store owner **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members
The Store owner 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 Store owner 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

[1.9] Use case: Update management permission for sub-manger

- **[Req: 4.6, Class: User, CNAME: UpdateManagerPermissions, Test:TestChnageManagerPermissions]**
- **Actor: Store Owner**
- **Precondition:** The user is owner of the store the sub mangers set is manages and the set is not empty.
- **Parameter:** Set of permissions
- **Actions:**
  1. **Store Owner** chooses sub-manger from an non empty sub mangers set for specific store.
  2. The **Store's Owner** sets a set of permissions for the selected sub manager.
  3. The set of permissions is valid:
     a) The sub manager gets the new permissions.
  4. Else
     a) The sub manager stays with the former permissions.

Action	Expected Result
The user 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	All the operations succeeded, management permissions were updated for the requested manager.

permissions, the sub manager does operations required each of the permissions from the set of permissions.	
The user 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 user 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.

[1.10] Use case: Remove management permission for sub-manger

- **[Req: 4.6, Class: User, CNAME: UpdateManagerPermissions, Test:TestChnageManagerPermissions]**
- **Actor:** Store Owner
- **Precondition:** The user owns the given store and sub manager is actual sub manager of the store.
- **Parameter:** Store and sub manager, and list of permissions to remove
- **Actions:**
  1. The Store's Owner asks to remove the list of permissions from the manager's permissions in the store
  2. The right management permissions removed from the former manager permissions for the specific store

Action	Expected Result
The user owns the store and the Store's Owner removes permissions from 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 user 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 user 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.

[1.11] Use case: Review store's stakeholders

- **[Req: 4.9, Class: Store, CNAME: GetStoreStaff]**
- **Actor:** **Store Owner**
- **Precondition:** The user is owner of the store
- **Parameter:** store id
- **Actions:**
  1. **Store Owner** asks to get information about staff working in the store
  2. System checks that the user is an owner of the store
  3. System checks with the store for each user connected to the store with any role
  4. System returns list of user ids and list of each user's permissions

Action	Expected Result
Store owner asks for information about store's staff	All correct information is provided
Store owner asks for information about a store he doesn't own	System informs that the user doesn't have the permissions to check this information
Store owner asks for information about a store that doesn't exist	System informs that there is no such store in the system

[1.12] Use case: Admin requests for all users history

- **[Req: 6.4, Class: UserManager, User, CNAME: AdminGetAllUserHistory, Test:TestAdminRequestsHistory]**
- **Actor:** **Admin**
- **Precondition:**
- **Parameter:**
- **Actions:**
  1. The admin requests to print the history of all user's purchase
  2. If The admin has ecommerce management permissions:
     a) The admin get the requested information
  3. Else:
     a) The admin get a message for inappropriate permissions.

Action	Expected Result
Admin asks to view all purchase history	The history is provided successfully.
Admin asks to view information about purchases of items that their prices are different now	The prices shown by the history are that same as they were in the actual transaction
Admin asks to view information about purchases of users that are no longer users in the system	System provides and preserves the information for all purchases

Admin asks to view information about purchases of items that were deleted from the system, or of stores that were closed and removed from the system	The transaction's information is preserved and saved as they were, and presented properly.

[1.13] Use case: Admin requests for store history

- **[Req: 6.4, Class: User Manager, User, CNAME: AdminGetStoreHistory, Test: TestAdminRequestsHistory]**
- **Actor: Admin**
- **Precondition:** Store exists
- **Parameter:** Store id
- **Actions:**
  1. The admin requests to print the history of a requested store id
  2. If the admin has ecommerce management permissions:
     a) The admin get the requested information
  3. Else:
     a) The admin get a message for inappropriate permissions.

Action	Expected Result
Admin asks to view information of store's history	The history is provided successfully.
Admin asks to view information about purchases of items that their prices are different now	The prices shown by the history are that same as they were in the actual transaction
Admin asks to view information about purchases of users that are no longer users in the system	System provides and preserves the information for all purchases
Admin asks to view information about purchases of items that were deleted from the system	The transaction's information is preserved and saved as they were, and presented properly.
Admins asks to view information about purchases from store that doesn't exist	System informs the admin the store doesn't exist

[1.14] Use case: Remove Co-owner from Store

- **[Req: 4.4, Class: User, CNAME:RemoveCoOwner, Test:TestAppointCoOwner]**
- **Actor: Owner**
- **Preconditions: Owner is the user to appoint Co-owner**
- **Parameters: User id and Store id**
- **Actions:**
  1. Owner requests to remove a Co-owner from the store.
  2. System requests User id and Store id
  3. Owner provides the requested information.
  4. System checks if Owner is the one to appoint Co-Owner to the store.
  5. If yes:
     a) System removes Co-Owner from store. Aswell as requests to remove all Co-owners he appointed.
  6. If no:

a) System returns an error message.

Action	Expected Result
Store owner 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 a Co-owner appointed by the Store owner	The system removes the co-owner from the store and updates the nominations to remove the co-owner and the owners he nominated
Store owner is logged to the system and provides identification of a store that doesn't exist	**System** informs **Store owner** that the **store** doesn't exist in his owned store repository
The Store owner is logged to the system and provides identification of a store he owns and an unknown user	System informs the **Store owner** that the **user is unknown to the system**

2. Items and stores:

[2.1] Use case: Gather information about store/product

- **[Req: 2.5, Class: StoreRepository, Store, CNAME: SearchForProducts, SearchForStore, Test:TestSearchforProductsOrStores]**
- **Actor:** **User**
- **Precondition:** User opened the system
- **Parameter:** String (presenting a product or a store)
- **Actions:**
  1. **User** can choose to search for product or store, by entering a string query to search
  2. **System** searches through all it's system database for:
     a) **Products** that its name matches the provided information
     b) **Stores** that its name matches the provided information.
  3. **System** presents all the matching stores/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.

[2.2] Use case: Search for products

- **[Req: 2.6, Class: StoreRepository, Store, CNAME: SearchForProduct, Test:TestSearchforProductsOrStores]**
- **Actor:** **User**
- **Precondition:** There are products in the system
- **Parameter:** search term
- **Actions:**
  1. **User** types in a string representing what he looks for
  2. **System** searches through all the **products** in the data base-in all stores- 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 list : price, category
  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**

3. User buying from stores:

[3.1] Use case: Save items in a shopping cart

- **[Req: 2.7, Class: User,Cart,Basket CNAME: AddItemToCart, Test:TestSaveItemToCart]**
- **Actor: User**
- **Precondition:** There are available items to buy
- **Parameter:** Items
- **Actions:**
  1. **User** can choose a **item** to buy
  2. If the **item** already exists in the shopping cart
     a) **System** updates the amount of the **item** appearances in the **shopping cart**
  3. Else
     a) **System** adds **item** representation of the real store item to the **shopping cart**

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

[3.2] Use case: View shopping cart

- **[Req: 2.8, Class: User, Cart, Basket , CNAME: GetCart, Test:TestViewCart]**
- **Actor: User**
- **Precondition:**
- **Parameter:**
- **Actions:**
  1. **User** 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.

[3.3] Use case: Edit shopping cart-

- **[Req: 2.8, Class: User, Cart, Basket, CNAME: EditCart, Test:TestEditCart]**
- **Actor:** User
- **Precondition:** Shopping cart not empty
- **Parameter:** Item to update and chosen change to make
- **Actions:**
  1. System presents the existing **shopping cart**
  2. User chooses an **item** from the **cart**
  3. User chooses changing the amount
     a) If the new requested amount is 0, the **product** I removed from the **cart**
     b) 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

[3.4] Use case: Purchase the whole cart

- **[Req: 2.9, Class: User, Cart, Transaction, CNAME: BuyWholeCart, Test:TestBuyCart]**
- **Actor:** User
- **Precondition:** Cart is not empty
- **Parameter:** Payment Info
- **Actions:**
  1. User chooses to buy all the products in the cart
  2. **System** asks for **Payment info**
  3. **User** provides **Payment info**
  4. System starts to perform new **transaction**
     a) For each **Basket** in the **Cart**, **system** uses **Calculate total price for basket** use case
     b) If all the calculation performed correctly system informs all the store to update item's stock as needed to buy all the items
        1. System summarizes all the prices and perform payment request to the external payment system

2. System waits for respond from the payment system
3. If the payment was ok
    1. System sends a supply request with the user's details to the external supply system
    2. System waits for the positive respond from the supply system
    3. If the respond from the supply system is ok
        1. System notifies the user that the purchase performed successfully
        2. System updates the transaction of both user and all the stores involved, adding a new purchase history record to the history records
        3. System clears user's cart an all baskets
    4. Else if supply system responds fail with supply
        1. System responds that the supply cannot be performed
        2. System asks the payment system to perform money refund
        3. System informs the user that the money was refunded
        4. System informs all the stores to perform items restore, with all the amounts that were taken from all the stores
4. Else if the payment system responds fail with payment
    1. System informs the user about fail with payment
    2. System informs all the stores to perform item restore, with all the amounts that were taken from all the stores

c) Else if one of the items is missing or one of the calculation went wrong- **Calculate total price for basket** use case failed

1. System informs the user that the purchase of the item cannot be performed

Action	Expected Result
All products in cart are in stock in all stores, payment info is valid, payment system and supply system reply good respond	System updates the total price according to discount strategies in store, sends payment request to the payment system, sends supply request to the supply system, informs the user and adds the purchase record to the history
Not all products in cart are in stock	System informs the user that one of the items in store are not in stock- not enough of it in store, and cancels the purchase

All products in cart are in stock, but payment system responds with fail respond	System informs the user, and restores all items to the stores
All products are in cart are in stock, payment system responds ok but supply system responds fail	System informs user, sends refund request to payment system and restores all items to stores

[3.5] Use case: Calculate total price for basket

- **[Req: 2.9, Class: User, Cart, Transaction, CNAME: GetCartPrice, Test:TestCartPrice]**
- **Actor:** User
- **Precondition:** Basket is not empty
- **Parameter:** Basket with items, User
- **Actions:**
  1. System runs check for every item in basket
     a) **System** checks that there is enough items for the requested amount in the store
     b) **System** checks that all items in basket correspond with the stores policy
     c) **System** runs all stores discount strategies for all items in basket and returns the minimum price of all of them
     d) **System** updates the basket's total price

Action	Expected Result
All products are in stock and valid with policies	System informs that all products are ok and updates basket's total price
Not all products are in stock	System informs that not all product are in stock and can’t perform purchase
Items in basket don't comply with store's policy	System informs that purchase cannot be made because of store's policy, and cancels

[3.6] Use case: Offer Purchase for a Product

- **[Req: 2.9, Class: User, Cart, Basket CNAME:AddBidToItem]**
- **Actor: User, Store Owner**
- **Precondition:** product is sold in offer purchase.
- **Parameter:** item, user
- **Actions:**
    1. **User** requests to purchase **item** sold in an offer purchase.
    2. **System** request **User** to submit an offer.
    3. **User** Provides an offer
    4. **System** send offer to **Store Owners**
    5. Each **Store Owner** chooses whether to accept or decline.
    6. If all **store owners** accepts:
        a) **System** adds **item** to **User's cart** and sets it's price to be the offer.
        b) Proceed with "Use case: Purchase whole cart" for **transaction**.
    7. If a **store owner** denies:
        a) **Item** is not added to **Users** cart.
        b) **System** sends a message to the **user** that his offer was declined.

Action	Expected Result
Product is a product the store sells in bid form. And the owners accept the first offer	The product is added to the user's cart under his offer price
Product is a product the store sells in bid form. And the all the owners but one accept the first offer	The product is not added
Product is a product the store sells in bid form. And all owners decline	The product is not added

4. Store Manage:

[4.1] Use case: Open a store

- **[Req: 3.2, Class: User,Store, CNAME: OpenStore, Test:TestOpenStore]**
- **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**
        a) **System** signs the **Member** as the founder of the **store**

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.

[4.2] Use case: Add new product to store

- **[Req: 4.1, Class: Store, CNAME:ItemsToStore, Test:TestAddNewItemToStore]**
- **Actor: Store owner**
- **Precondition:**
  1. **Store owner** is logged in to the **system**
  2. **Store owner** 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. Store owner asks to add a new product to a store he owns
  2. The system asks for store identification
  3. Store owner enters store id
  4. System asks for product's details
  5. **Store owner** 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 store
     a) System informs Store owner and seller that a product already exists
  8. Else
     a) System adds the product to the store's products resource
     b) System updates product's current stock to 0
     c) System informs the Store owner and seller that the product was added successfully

Action	Expected Result
Store owner and seller 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

Store owner and seller is logged to the system and provides identification of a store that doesn't exist	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller is logged to the system and provides identification of a store that he doesn't own	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller 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 **Store owner and seller that the products he asks to add already exits in the store**
The Store owner and seller is not logged to the system	Message indicates that managing stock is only allowed to logged in existing members
The Store owner and seller 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 **Store owner and seller** that the product's details are invalid

[4.3] Use case: Remove product from store

- **[Req: 4.1, Class: Store, CNAME: ItemsToStore, Test:TestRemoveItemFromStore]**
- **Actor: Store owner**
- **Precondition:**
  1. **Store owner** is logged in to the **system**
  2. **Store owner** is an owner of an existing store
- **Parameter:** Store identification and product id
- **Actions:**
  1. Store owner asks to remove a product from a store he owns
  2. The system asks for store identification
  3. Store owner and seller enters store id
  4. System asks for product's id
  5. **Store owner** 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
     a) System deletes product from the store resources
     b) System informs Store owner and seller that a product was deleted successfully from the store
  8. Else
     a) System informs the **Store owner** that the product doesn't exists in the store

Action	Expected Result
Action: Store owner and seller **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

Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **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 **Store owner and seller that the products he asks to add doesn't exists in the store**
The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members

[4.4] Use case: Update product stock- add items

- **[Req: 4.1, Class: Store, CNAME: UpdateStockAdd , Test:TestEditItemInStore]**
- **Actor: Store owner**
- **Precondition:**
  1. **Store owner** is logged in to the **system**
  2. **Store owner** is an owner of an existing store
- **Parameter:** Store identification, product id, amount
- **Actions:**
  1. Store owner and seller asks to add product items to store's stock
  2. The system asks for store identification, product id and amount
  3. Store owner enters store id
  4. **Store owner** provides required details- store id, product id, amount
  5. System checks if such a store exists in user's store repository
  6. If exists
     a) System checks if such product exists in the store
     b) If exists
        1. System updates the product's current stock to the new amount
     c) Else
        1. Store asks if the **Store owner** wants to add the new product to the store
        2. If **Store owner** choose to add the product
           1. **Store owner** start **Add new product to store** use case
           2. System updates product's stock to the new amount

Action	Expected Result
Store owner and seller **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: Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **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 **Store owner and seller** that the products he asks to add doesn't exists in the store
The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members
The Store owner and seller **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

[4.5] Use case: Update product stock- subtract items

- **[Req: 4.1, Class: Store, CNAME: UpdateStockSub Test:TestEditItemInStore]**
- **Actor: Store owner**
- **Precondition:**
  1. **Store owner** is logged in to the **system**
  2. **Store owner** is an owner of an existing store
- **Parameter:** Store identification, product id, amount
- **Actions:**
  1. Store owner and seller asks to subtract product items from store's stock
  2. The system asks for store identification, product id and amount
  3. **Store owner** provides required details- store id, product id, amount
  4. System checks if such a store exists in user's store repository
  5. If exists
     a) System checks if such product exists in the store
     b) If exists
        1. System updates the product's current stock to the new amount
     c) Else
        1. System informs the **Store owner** that such product doesn't exist in the system

Action	Expected Result
Store owner and seller **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
Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **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 **Store owner and seller** that the products he asks to add doesn't exists in the store
The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members
The Store owner and seller **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

[4.6] Use case: Update existing product's details

[Req: 4.1, Class: MarketFacade, CNAME: EditItemToStore Test:TestEditItemInStore]

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

Action	Expected Result
Store owner and seller **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

Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **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 **Store owner and seller** that the products he asks to add doesn't exists in the store
The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members
The Store owner and seller **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

[4.7] Use case: Add buying strategy to store's policy

[Req:4.2, Class: Store, CNAME:AddRuleToStorePolicy, Test:TestRulesAndDiscounts]

- **Actor: Store owner**
- **Precondition:**
  1. **Store owner** is logged in to the **system**
  2. **Store owner** is an owner of an existing store
- **Parameter:** store id and buying strategy
- **Actions:**
  1. **User asks to add buying strategy to store**
  2. **System requests the store id and strategy**
  3. **User provides required information**
  4. **If strategy already exists**
     a) **System doesn't change it**
  5. **Else**
     a) **System adds new strategy to store's policy**

Action	Expected Result
Store owner and seller **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
Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository

The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store he owns and an unknown buying strategy**	System informs the **Store owner and seller** that the buying strategy is unknown
The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members

[4.8] Use case: Update buying strategy to store's policy

[**Req:4.2, Class: Store, Test:TestRulesAndDiscounts]**

- **Actor: Store owner and seller**
- **Precondition:**
  1. **Store owner and seller** is logged in to the **system**
  2. **Store owner and seller** is an owner of an existing store
- **Parameter:** store id and buying strategy
- **Actions:**
  1. **User asks to update buying strategy to store**
  2. **System requests the store id and strategy**
  3. **User provides required information**
  4. **System locates store and check if strategy exists in the store's policy**
  5. **System asks whether to delete or update existing strategy**
     a) **If user chooses to update**
        0. **System update the strategy in the store with the new strategy provided**
     b) **If user chooses to delete**
        0. **System removes strategy from store's policy**

Action	Expected Result
Store owner and seller **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
Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store he owns and an unknown buying strategy**	System informs the **Store owner and seller** that the buying strategy is unknown

The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members
The Store owner and seller **is logged to the system and provides identification of a store he owns and a strategy that doesn't exist**	System informs the **Store owner and seller** that the buying strategy doesn't exist and encourages him to use **Add new strategy to store's policy** use case

[4.9] Use case: Add allowed discounts to store's policy

[Req:4.2, Class: Store, CNAME: AddDiscountToStore, Test:TestRulesAndDiscounts]

- **Actor: Store owner and seller**
- **Precondition:**
  1. **Store owner and seller** is logged in to the **system**
  2. **Store owner and seller** 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**
     a) **System doesn't change it**
  5. **Else**
     a) **System adds new discount type to store's policy**

Action	Expected Result
Store owner and seller **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
Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store he owns and an unknown** discount type	System informs the **Store owner and seller** that the **discount type** is unknown
The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members

[4.10] Use case: Update allowed discounts in store's policy

[Req:4.2, Class: Store, Test:TestRulesAndDiscounts]

- **Actor: Store owner and seller**
- **Precondition:**
  1. **Store owner and seller** is logged in to the **system**
  2. **Store owner and seller** 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**
  5. **System asks whether to delete or update existing** discount type
     a) **If user chooses to update**
        0. **System update the** discount type **in the store's policy with the new** discount type **provided**
     b) **If user chooses to delete**
        0. **System removes** discount type **from store's policy**

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

[4.11] Use case: View store's policy

[Req:4.2, Class: Store, CNAME: GetStorePolicy, Test:TestRulesAndDiscounts]

- **Actor: Store owner and seller**
- **Precondition:**
  1. **Store owner and seller** is logged in to the **system**
  2. **Store owner and seller** 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. **System presents store policy**

Action	Expected Result
Store owner and seller **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
Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is not logged to the system**	Message indicates that managing store's policy is only allowed to logged in existing members

[4.12] Use case: Add buying strategy to store's product

[Req:4.2, Class: Store, CNAME: GetStoreDiscounts, Test:TestRulesAndDiscounts]

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

Action	Expected Result
Store owner and seller **is logged to the system and provides an identification of an existing store that he owns, valid buying strategy that exists in store's policy and an id of a product exists in the store**	The system add buying strategy to the product in the store
Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn’t exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn’t own**	**System** informs **Store owner and seller** that the **store** doesn’t exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store he owns and an unknown** buying strategy	System informs the **Store owner and seller** that the **buying strategy** is unknown
The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members
The Store owner and seller **is logged to the system and provides identification of a store he owns and a known buying strategy and a product id that doesn’t exist in store**	Message indicates that managing product doesn't exist in store
The Store owner and seller **is logged to the system and provides identification of a store he owns and a known buying strategy that doesn't exist in store's policy and a product id that exists in store**	Message indicates that strategy isn't allowed in store

[4.13] Use case: Update buying strategy to store's product

[Req:4.2, Class: Store, Test:TestRulesAndDiscounts]

- **Actor: Store owner and seller**
- **Precondition:**
  1. **Store owner and seller** is logged in to the **system**
  2. **Store owner and seller** is an owner of an existing store
- **Parameter:** store id, product id and buying strategy
- **Actions:**
  1. **User asks to update buying strategy to product**
  2. **System requests the store id, buying strategy and product id**
  3. **User provides required information**
  4. **If store exists, buying strategy valid and exists and product exists in store**
     a) **System asks whether to delete or update**
        0. **If update, System check if requested buying strategy exists in store's policy**
           0. **If exits in policy- system update existing buying strategy of product in store**

1. **If delete- System removes buying strategy from product in store**

Action	Expected Result
Store owner and seller **is logged to the system and provides an identification of an existing store that he owns, buying strategy that exist for the product in the store and an id of a product exists in the store**	The system update buying strategy to the product in the store
Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store he owns and an unknown** buying strategy	System informs the **Store owner and seller** that the **buying strategy** is unknown
The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members
The Store owner and seller **is logged to the system and provides identification of a store he owns and a known buying strategy and a product id that doesn't exist in store**	Message indicates that managing product doesn't exist in store
The Store owner and seller **is logged to the system and provides identification of a store he owns and a known buying strategy that doesn't exist in store's policy and a product id that exists in store**	Message indicates that strategy isn't allowed in store

[4.14] Use case: View product's buying strategies

[**Req:4.2, Class: Store, Test:TestRulesAndDiscounts]**

- **Actor: Store owner and seller**
- **Precondition:**
  1. **Store owner and seller** is logged in to the **system**
  2. **Store owner and seller** is an owner of an existing store
- **Parameter:** store id, product id
- **Actions:**
  1. **User asks to view buying strategy of product in store**
  2. **System requests the store id and product id**
  3. **User provides required information**
  4. **If store exists and product exists in store**
     0. **System provides all exiting buying strategies of the product in the store**

Action	Expected Result
Store owner and seller **is logged to the system and provides an identification of an existing store that he owns and an id of a product exists in the store**	The system provides all existing buying strategies of the product in the store
Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store he owns and a product id that doesn't exist in the store**	System informs the **Store owner and seller** that the **product doesn't exist in the store**
The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members

[4.15] Use case: Add discount to product

[Req:4.2, Class: Store, Test:TestRulesAndDiscounts]

- **Actor: Store owner and seller**
- **Precondition:**
    1. **Store owner and seller** is logged in to the **system**
    2. **Store owner and seller** is an owner of an existing store
- **Parameter:** store id, product id and discount
- **Actions:**
    1. **User asks to add discounts to product**
    2. **System requests the store id, discount type and discount details and product id**
    3. **User provides required information**
    4. **If store exists, discount type and details valid and exists and product exists in store**
        a) **System checks if discount is valid in store's policy**
        b) **If valid- System adds discount to product in store**

Action	Expected Result
Store owner and seller **is logged to the system and provides an identification of an existing store that he owns, valid discount type and details that valid with system in general and with store's policy, and an id of a product exists in the store**	The system adds the discount to the product int the store
Store owner and seller **is logged to the system and provides identification of a store that doesn't exist**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository
The Store owner and seller **is logged to the system and provides identification of a store that he doesn't own**	**System** informs **Store owner and seller** that the **store** doesn't exist in his owned store repository

The Store owner and seller **is logged to the system and provides identification of a store he owns and an unknown** discount type	System informs the **Store owner and seller** that the **discount type** is unknown
The Store owner and seller **is not logged to the system**	Message indicates that managing stock is only allowed to logged in existing members
The Store owner and seller **is logged to the system and provides identification of a store he owns and a valid discount and a product id that doesn't exist in store**	Message indicates that managing product doesn't exist in store
The Store owner and seller **is logged to the system and provides identification of a store he owns and a valid discount that is not valid with store policy, and a product id that exists in store**	Message indicates that discount isn't allowed in store

[4.19] Use case: Store Owner get purchase history of a store

- **[Req: 4.11, Class: MarketFacade, CNAME: GetStoreHistory, Test:TestGetPurchaseHistoryOfStore]**
- **Actor: Store Owner**
- **Precondition: User is owner of the store**
- **Parameter:** StoreID
- **Actions:**
  1. The User requests to print the history of purchase for specific sore
  2. If The User is logged, the storeID belongs to existing store in the system ownership permissions for the specific store
     a) The user get the requested information
  3. Else
     a) The user get a message for the specific problem.

Action	Expected Result
The user 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 user 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 user 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