The color map : System - cyan . Guest – light green . member – purple . store owner- pink . system manager – light orange (?)

Notice: we add the necessary diagrams in the end of the file and the name of the use cases that we split.

Use case name and number	Description or diagram.	Actors	Precondition	Parameters	Post Condition	Main Scenario:	Alternative Scenarios:
I.1 – Market activation	A user activates the market system for the first time, and becomes the first system manager.	user (and he will be also the system manager).	system must be off.	initial authentication details (username and password) and details about one payment service and one supply service.	system must be on.	1. user: turn on the system. 2. system: asks the user for initial authentication details. 3. user: enters its (new) authentication details. 4. system: verifies the validity of the given details. 5. system: opens the system, with one member and system manager that both represent the user, with the given external services. 6. system: responses a success to the user. 7. system: start serving users.	- User's initial authentication details not verified successfully (4), system asks for the details again. - Payment service or supply service are not verified successfully (4), the system asks for the details again. - System fails preparation (5), the system cancels the action and waits for System Manager action.
I.2 - change/switch/ add an external service	A system manager changes/switches/add s an external service	system manager	system must be on, the system manager must be login	details of the new service (payment or stock)	the system must continue to work the same as before the change of the service, with the change of the service.	1. system manager: chooses the service he wants to edit/add/switch 2. system manager: enters the details of the service 3. system: authenticates the details the user entered and his permission. 4. system: changes/add/switches the service for the new one 5. system: send positive response to the user	- User's not have permission to do this act successfully (3) - service details are incorrect and requested to enter again.

I.3 : Using the services of an external payment system	the system contacts a payment system which the market is familiar with and tries to perform a payment and receive payment validation	System	the system is activated	details of the transaction	the system must get payment validation, and must continue to work the same as before the change of the service	1. the system contacts the external payment system with transaction details to perform payment 2. the external payment sends back a positive validation for the transaction the system tried to perform	- the transaction details the system sent are invalid - the payment service returned a negative confirmation for the payment the system tried to do because of problem with the transaction (not because of the details)
I.4 - Using the services of an external supply service:	The system contacts a supply system which the market is familiar with and asks for approval of a valid supply	System	System is activated	details of the supply	the system must get approval of a valid supply, and continue to work the same as before the request of service	1. system: contacts an external supply system it is familiar with and sending it the details of the required package and client information 2. external service: sends back a positive answer that the request was approved	- the supply details the system sent are invalid - The external service sends back a negative answer (2) that the request was declined, and the reason for the decline of the request.
I.5.1 real time notifications to store owner	System must give real time notifications to logged in store owners in one of the scenarios: 1. A product in one of the store owner's stores is bought. 2. One of the store owner's stores is being opened. 3. One of the store owner's stores is being closed. 4. One of the store appointments as a store owner is being removed	A store owner	System must be active, and the store owner is logged in.	None	the store owner must get the notification, and system continue to work the same as before the request of service.	1. One of the scenarios that described earlier happens. 2. The system notifies the store owner about the event that occurred.	If the user not logged in the system will act like use case I.6
I.5.2 real time notifications to member	System must give real time notifications to logged in member in case of receiving message	A member	System must be active, and the member is logged in.	None	the member must get the notification, and system continue to work the same as before the request of service.	1. The member receive message. 2. The system notifies the member about the event that occurred.	If the user not logged in the system will act like use case I.6
I.6 - Member notifications show up when logging in:	A member logs in to the system, and the notifications that were meant for them during	Member	System must be active, Member must be logged in	None	System must continue to work normally as it should after any	1. member: logs in to the system. 2. system: shows up all notifications of the member to them.	- No new notifications to show (2) so the system show to the member " There are no new notifications".

	the time they were logged off are shown.				login, without considering the notifications.		
II.1.1 – Guest Entrance	A user enters the system as a guest, the system defines him as guest and assigns him his own shopping-cart.	User	System server is turned on	None	the user is connected to the system as a guest	1. User: requests to enter the system 2. System: accepts the user 3. System: defines the user as a guest 4. System: creates a shopping cart assigned to the user 5. System: alerts the user with a response that the operation succeeded	If the system loses its connection (whether because of disconnection of the user, or any other reason) with the user during steps 2-5, it cancels the previous actions and ends the communication.
II.1.2 – Guest Disconnection	A guest disconnects the system, the system removes his assigned shopping cart.	Guest	System server is turned on, the guest user is connected to the system	None	the user is no longer connected to the system a guest	1. Guest: requests the system to log out 2. System: Unassigns the shopping cart of the user and deletes it 3. System: Closes the connection with the user	If the user closes his connection during this action, if it happens before (2) the system deletes his shopping cart. The system then skips step 3 and acts as if the request succeeded.
II.1.3 – Guest Registration	A guest registers the system giving unique authentication details, later the system recognizes him as a member when logging in using these details.	Guest	System server is turned on, the guest user is connected to the system, there is no member account with the specified authentication details registered to the system.	Authentication Details	A member account with the given authentication details is registered to the system and can log in using them.	1. Guest: requests the system to register, specifying authentication details 2. System: verifies the authentication details and checks that they are unique among the registered members of the system 3. System: Stores a new member with the given authentication details 4. System: alerts the user the action has succeeded	If the user closes his connection during this session – the system undoes the actions that were done previously and does not register the user. If the details given by the user are not unique – the system alerts the user the operation has failed.
II.1.4 – Guest Login as a Member	A guest who has registered the system as a member, can use his unique authentication details to log into the system as a member.	Guest	System server is turned on, the guest user is connected to the system, the given authentication details exist in the system	Authentication Details	user state is member.	1. Guest: requests the system to login, specifying authentication details 2. System: verifies that the specified authentication details exist in the stored member details. 3. System: changes the user state from a guest to a member 4. System: alerts the user the action has succeeded	If the user closes his connection during this session – the system undoes the actions that were done previously and does not log in the user. If the details given by the user do not exist in the system – the system alerts the user that the operation has failed.
II.2.1.A – Guest Information Receival about Stores	A guest can access the information about all the active stores in the system.	Guest	System server is turned on	None	None	1. Guest: requests information about the stores in the system 2. System: returns to the user a list of details of all the active stores in the system	None

						3. System: alerts the user the action has succeeded	
II.2.1.B – Guest Information Receival about Products in a Store	A guest can access the information about all the products in an active store in the system.	Guest	System server is turned on, store identifier exists in the system	Store identifier	None	1. Guest: requests information about the products in a specific store, and specifies its identifier 2. System: verifies that there exists a store with this identifier in the system 3. System: returns to the user a list of details all the products in the desired store 4. alerts the user the action has succeeded	if the verification in (2) fails the system alerts the user the action has failed.
II.2.2 – Guest Search Products	A guest can search product regardless of a specific store by mandatory fields (name, category, or keywords) and filter the results according to several attributes.	Guest	System server is turned on	Mandatory search fields, additional filtering attributes	None	1. Guest: request details of products relevant to the specified search fields and filtering attributes 2. System: verifies that the guest filled at least one of the mandatory fields 3. System: returns to the user a list of details of all the products relevant to the given search fields and filtering attributes. 4. alerts the user the action has succeeded	if the verification in (2) fails – alert the user the action has failed
II.2.3 – Guest Add Products	A guest can add products to his store bag, as a part of his own shopping cart.	Guest	System server is turned on, store id exists in the system, product id is available in the desired store, count is a positive integer.	store id, product id, count	the guest's shopping cart contains a store bag of the specified store which contains the specified amount of the specified product	1. Guest: request to add product from a store, also specifying the amount of such products 2. System: verifies that - store id exists in the system, product id is available in the desired store, count is a positive integer and does not exceed the number of products in the store stock. 3. System: adds the amount of the desired product to the guest's store bag, and updates his own shopping cart respectively. 4. alerts the user the action has succeeded	if the verification in (2) fails – the system alerts the user the action has failed
II.2.4.A – Guest View Cart Contents	A guest can watch his shopping cart contents.	Guest	System server is turned on	None	None	1. Guest: requests to watch his cart contents 2. System: returns the user a list of his car contents - its products details (including the stores they are from). 3. alerts the user the action has succeeded	None
II.2.4.B – Guest Remove	A guest can remove products from his	Guest	System server is turned on, store id	store id, product id	the guest's shopping cart no	1. Guest: request to remove product from a store bag in his own shopping cart	if the verification in (2) fails – the system alerts the user the action has failed

Product from Cart	store bag, as a part of his own shopping cart.		exists in the system, product id is available in the desired store.		longer contains the specified product in the specified store's bag.	2. System: verifies that – a relevant store bag exists in the user's shopping cart and some products with the specified id appears in the desired store bag. 3. System: removes the specified product from the specified store bag in the guest's shopping cart. 4. alerts the user the action has succeeded	
II.2.4.C – Guest Update Product Amount in Cart	A guest can change the amount of a specific product in his store bag, as a part of his own shopping cart.	Guest	System server is turned on, store id exists in the system, product id is available in the desired store, count is positive integer.	store id, product id, count	the guest's shopping contains the specified amount of the specified product in the specified store's bag.	1. Guest: request to change the amount of a specific product from a store bag in his own shopping cart 2. System: verifies that – a relevant store bag exists in the user's shopping cart, count is positive integer and some products with the specified id appears in the desired store bag. 3. System: sets the amount of the specified product in the specified store bag in the guest's shopping cart to be count. 4. alerts the user the action has succeeded	if the verification in (2) fails – the system alerts the user the action has failed
II.2.5 – Guest Purchase Cart Contents	A guest can purchase his cart contents if all of them are available.	Guest	System server is turned on, guest's shopping cart is nonempty, all the cart contents are available in the relevant stores	delivery details, payment method, payment details	the guest's shopping cart becomes empty.	1. Guest: request to purchase his own shopping cart contents 2. System: verifies that that guest's shopping cart is nonempty and all the cart contents are available in the relevant stores. 3. System: verifies the payment method and details. 4. System: makes the purchase. 5. System:use case i.6 : sends the delivery details to the delivery company. 6. System: charges the guest according to the payment method and details. 7. System: stores the order details. 8. System: sends the guest a receipt 9. System: cleans the guest's cart 10. alerts the user the action has succeeded	if any of the verifications (2) or (3) fail – the system alerts the user the action has failed
Member use cases II.2.1-II.2.5	The members are able to do II.2.1-II.2.5 exactly the same way as guests except that	Member	the user must be logged in.				

	now new precondition.						
II.1.2.2 - Member disconnection	A Member can disconnect from the system.	member	System must be active, member must be logged in.	None	System must continue to work normally as it should after any disconnection, member must be logged out.	1. member: send a logout request. 2. system: saves the member's cart. 3. system: disconnects the member and closes the connection.	The member closes the connection (1) without ask to disconnect, the system acts as a disconnection request has been received.
II.3.1 - Member logout	A Member can logout from the system.	member	System must be active, member must be logged in.	None	System must continue to work normally as it should after any logout, member must be logged out.	1. member: send a logout request. 2. system: saves the member's cart. 3. system: logs out the member. 4. system: change the user state to be guest. 5. system: sends a successful response with a new empty cart.	The member closes the connection (1) without ask to logout, the system acts as a disconnection request has been received
II.3.2 - Create new store	A Member can create a new store and become the first store owner.	member	System must be active, member must be logged in	store details	System must continue to work normally, with new store and the member as the owner of that new store.	1. member: sends request for creating new store with store information. 2. system: checks the validity of the new store details. 3. system: creates a new store with given details. 4. system: sets the member to be the founder of the store. 5. system: sends a successful response with store id.	The member gives not valid details (one or more) (2), the system returns error message.
II.4.1.a - Adding new product to the store	The store owner can add product to the store.	Store owner	System must be active, store must be active, store owner must be logged in.	new product details	System must continue to work normally, with new product in the product catalog given store.	1. owner: sends request for adding new product to the store. 2. system: checks the product details validity. 3. system: add the product to store collection using the given details, with initial amount of 0.	The parameters that were given are not valid details(can be for some reasons), the system returns error message with the specific information.

						4. system: sends a successful response and product id.	
II.4.1.b - Removing products from the store	The store owner can remove product from the store.	Store owner	System must be active, store must be active, store owner must be logged in, product id must be of an active product in store.	product id	System must continue to work normally, without the product in the product catalog given store	1. owner: sends request for removing existing product from the store. 2. system: checks the id validity. 3. system: remove product from the store. 4. system: sends a successful response.	The parameter that given is not valid id (the product doesn't exist), the system returns error message with the specific information.
II.4.1.c - Changing product's quantity in inventory	The store owner can change the quantity of product from the store.	Store owner	System must be active, store must be active, store owner must be logged in, product id must be of an active product in the store.	product id, new quantity.	System must continue to work normally, with product quantity equals to the quantity that given	1. owner: sends request for updating product's quantity in the store. 2. system: checks product's quantity and id validity. 3. system: update product's quantity in store inventory. 4. system: sends a successful response.	The parameter that given is not valid id (the product doesn't exist), the system returns error message with the specific information. The parameter that given is not valid quantity (negative), the system returns error message with the specific information
II.4.1.d - Changing product's details in the store	The store owner can change the details (such as price), of product from the store	Store owner	System must be active, store must be active, store owner must be logged in, product id must be of an active product in the store.	product id, new product's details.	System must continue to work normally, with product details equals to the details that given.	1. owner: sends request for updating product's details in the store. 2. system: checks params validity. 3. system: update product's details in store. system: sends a successful response	The parameter that given is not valid id, the system returns error message with the specific information. The parameter that given is not valid details, the system returns error message with the specific information
II.4.2 - Change type of purchases and discount policy	The store owner can change the purchases and discount policy of the store.	Store owner	System must be active, store must be active, store owner must be logged in	store id, purchases and discount policy	New policy should apply	1. owner: sends request for changing policy in the store. 2. system: sets new policy for the store. 3. system: sends a successful response.	The parameter that given is not valid id, the system returns error message with the specific information.
II.4.4 - Make store owner	The store owner can make other member	Store owner	System must be active, store must be	member id	Store owners hierarchy is a tree (no cycles)	1. owner: sends request for making new owner in the store.	The member does not exist (2), the system cancels the action.

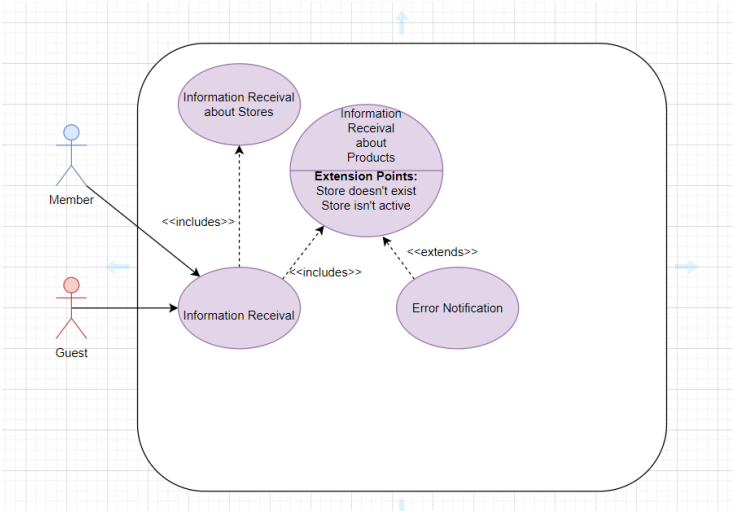
	to be also the store owner.		active, store owner must be logged in			2. system: verify member exists and not already owner. 3. system: sets new owner under the actor's (owner) in the hierarchy. 4. system: sends a successful response.	The member is already an owner of the store (2), the system cancels the action.
II.4.6 - Make store manager	The store owner can make other member to be also the store owner.	Store owner	System must be active, store must be active, store owner must be logged in.	member id.	The member role is assigned to be new store manager, and the assigner is the actor.	1. owner: sends request for making new manager in the store. 2. system: verify member exists and not already manager. 3. system: sets the new manager with the owner as appointor. 4. system: sends a successful response.	The member does not exist (2), the system cancels the action. The member is already a manager of the store (2), the system cancels the action.
II.4.7 – Owner Update Store Manager Permission	Store can edit the permissions each his store's managers has.	Store owner	System server is turned on, store owner is logged in, store is active, the owner appointed this manager.	store id, manager's id, new permissions	the manager's new permissions are updated.	1. Owner: requests to edit store manager's permission 2. System: verifies that the store id exists in the system and is active. 3. System: verifies that the owner owns this store. 4. System: verifies that the manager manages this store. 5. System: verifies that the manager was appointed by this owner. 6. System: edits the manager permissions respectively. 7. System: alerts the owner the action has succeeded	if any of the verifications (2) - (5) fail, the system alerts the user the action has failed.
II.4.9 – Founder Close Store	A founder can make his store inactive.	Founder -the first store owner	System server is turned on, founder is logged in, the store owners and managers are available in the system	founder's member identifier, the identifier of the store desired to be closed	the products in the newly closed store won't appear in the product search results, the store owners and managers are notified about their newly closed store	1. Founder: requests to close one of his stores 2. System: verifies that the store id exists in the system. 3. System: verifies that the founder id exists. 4. System: verifies that the store was indeed founded by this founder. 5. System: verifies that the store is active. 6. System: closes the store.	if any of the verifications (2) - (5) fail, the system alerts the user the action has failed.

						7. System: notify the store owners and managers that this store was closed 8. System: alerts the founder the action has succeeded	
II.4.11.A- Get Roles Holders Details	Store owner can access the information about the roles holders in his store	Store owner	System server is turned on, store owner is logged in	store owner's member id, store id	None	1. Owner: requests to get information about role holders in his store 2. System: verifies that the store id exists in the system. 3. System: verifies that the owner owns this store. 4. System: returns a list with data about each role holder in this shop to the owner 5. alerts the owner the action has succeeded	if any of the verifications (2) - (3) fail, the system alerts the user the action has failed.
II.4.11.B - Get Managers' Permissions	Store owner can access the information about the permissions his store's managers have.	Store owner	System server is turned on, store owner is logged in	store owner's member id, store id	None	1. Owner: requests to get managers' permissions in his store 2. System: verifies that the store id exists in the system. 3. System: verifies that the owner owns this store. 4. System: returns a list with the permissions for each manager in this shop to the owner 5. System: alerts the owner the action has succeeded	if any of the verifications (2) - (3) fail, the system alerts the user the action has failed.
II.4.13 - Owner Get Purchase History in the Store	Store owner can get the purchase history in his stores.	Store owner	System server is turned on, store owner is logged in	store owner's member id, store id	None	1. Owner: requests to get the purchase history in one of his stores 2. System: verifies that the store id exists in the system. 3. System: verifies that the owner owns this store. 4. System: returns a list with the all the purchases happened in this store to the owner 5. System: alerts the owner the action has succeeded	if any of the verifications (2) - (3) fail, the system alerts the user the action has failed.
6.4 - Get the purchase history of a store/buyer by	The system manager can get the purchase history of a store/buyer by a system manager	System manager	System must be active, system manager must be logged in	Admin id, store/buyer id	None	1.The admin requests the purchase history of the store/buyer and inputs the required credentials(admin id) and the information regarding the store/buyer(store/buyer id).	The admin id entered by the system manager is for an admin not currently existing in the system so the action can't

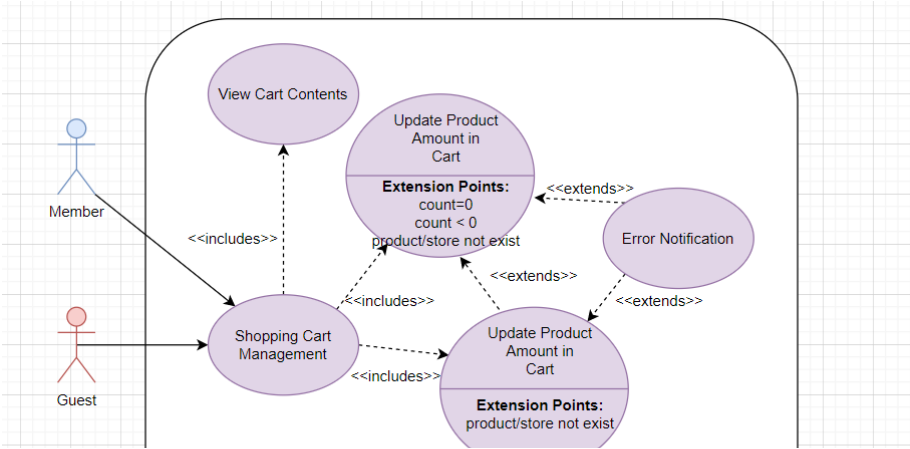
a system manager						2.The system verifies that the admin of the given id exists in the system, and that the store/buyer id corresponds to an existing store/buyer in the system. 3.The system outputs the purchase history of the entered store/buyer	be performed so an error message is returned via a notification. The store/buyer id does not exist in the system so the action can't be performed so an error message is returned via a notification.

:Use Cases Diagrams

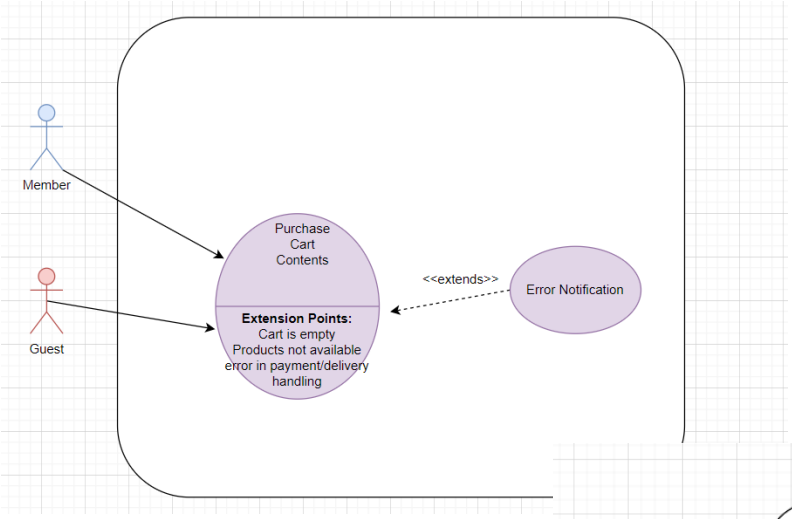
Use Case II.2.1 - Info request:



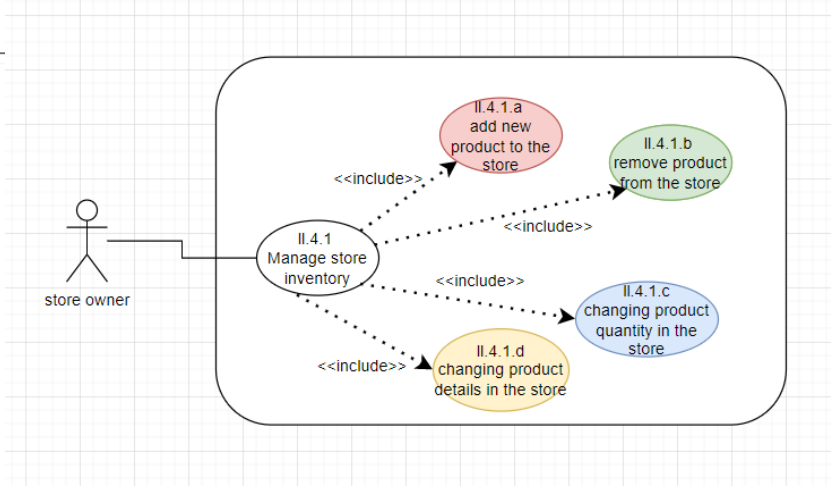
Use Case II.2.4 - Manage cart:



Use Case II.2.5 - Purchase cart content:



Use Case II.4.1 - Inventory management



We also split the following use cases to sub use case in order to avoid complex use cases:
l.5 real time notification for member and owner split into l.5.1 and l.5.2

II.2.1 Info receival split into ll.2.1.a and ll.2.1.b

ll.2.4 shopping cart management split into ll.2.4.a ll.2.4.b ll.2.4.c

ll.4.1 inventory management split into ll.4.1.a ll.4.1.b ll.4.1.c ll.4.1.d

ll.4.11 get store roles details split into ll.4.11a ll.4.11b