Use-Cases - 2020

Content

1.System

Use case name	#1.1 System initiation
Actors	SystemInitializer, TradeSystem
Pre-conditions	Admin details.
Post-conditions	1. Admin exists with admin username and password 2. 1 UserManager was created 3. 1 Admin was created 4. 1 Purchase was created 5. There is communications with the Shipment system 6. There is communications with the Payment system
Input Parameters	1. Admin details.
Main success scenario	1. **SystemInitializer**: creates new TradeSystem 2. **TradeSystem**: creates new UserManager 3. **TradeSystem:** creates new User with input parameters 4. **TradeSystem:** makes the created User (3) to Admin 5. **TradeSystem**: creates new Purchase 6. **TradeSystem**: creates new SystemManager 7. **SystemInitializer**: make a connection with Shipment system 8. **SystemInitializer**: make a connection with PaymentCollection system
Alternatives/Extensions	
Trigger	SystemInitializer get the order to open the trade system

Action	Data	Expected Result
#1.1 - (AT1) System initiation (external systems communication)	Insert Valid communication details with Payment and Shipment external systems	Success: Trading system activated. Allows transactions including payment and shipping
	Insert invalid communication details with Payment and Shipment external systems	Failure: receive an error message that the system could not be open due to communication with the external systems.

Action	Data	Expected Result
#1.1 - (AT2) System initiation (Creates Admin user)	Valid User details	Success: The User is created and then becomes Admin
	Invalid User details	Failure: No User and Admin are created
	Missing User details	Failure: No User and Admin are created

2.Guest User

Use case name	#2.2 Registration to the system
Actors	User
Pre-conditions	1. The current situation is according to Use-Case 2.1
Post-conditions	1. The User successfully registered to the system
Input Parameters	Registration details
Main success scenario	1.**User:** Requests to register to the system with the details provided. 2. **TradeSystem:** If the information provided by the User is correct and complete, the system registers the user.
Alternatives/Extensions	~2. Registration failed
Trigger	User requests to register to the system.

Action	Data	Expected Result
#2.2 - (AT1) Registration to the system	Valid registration details that do not yet exist in the system.	Success: the registration successfully completed. The user is registered with his provided details.
	Missing registration details.	Failure: The system registration was unsuccessful, no new user was added to the system
	Existing user details.	Failure: The system registration was unsuccessful, no new user was added to the system
	Invalid user details	Failure: The system registration was unsuccessful, no new user was added to the system

Use case name	#2.3 User Login to the system
Actors	User
Pre-conditions	1. The current situation is according to Use-Case 2.1
Post-conditions	1. The user is no longer in guest mode
Input Parameters	Login details.
Main success scenario	1.**User:** Requests to login to the system with the details provided. 2.**TradeSystem:** If the details provided by the user are complete, proper and existing in the system, the user is logged in
Alternatives/Extensions	~2. Login failed
Trigger	GuestUser requests to login to the system

Action	Data	Expected Result
#2.3 - (AT1) User Login to the system	Valid login details that exist in the system.	Success: The system connects the user. The user is no longer in guest mode.
	Missing registration details.	Failure: User should provide all login details
	User details that does not exist in the system	Failure: The login was unsuccessful, The user stays in guest mode.
	Invalid user details.	Failure: The login was unsuccessful, The user stays in guest mode.

Use case name	#2.4 View information of store and the products of the store
Actors	User
Pre-conditions	1. The current situation is according to Use-Case 2.1
Post-conditions	None.
Input Parameters	Requested store details
Main success scenario	1.**User**: provides the requested store details. 3. **TradeSystem**: If the requested store exists in the system 3.a **TradeSystem:** Returns information about the store and its products.
Alternatives/Extensions	~3. The system will return an error / message about incorrect or missing existing store information.
Trigger	A User wants to see details about a store and its products.

Action	Data	Expected Result
#2.4 (AT1) - Search a store and view its details and products	Details of a store that exists in the system.	Success: The system returns store details and store product details.
	Missing store details.	Failure: User should provide all store details.
	User details that do not exist in the system.	Failure: The store search failed, does not return results.
	Invalid user details.	Failure: Returns error object

*System sequence diagram of 2.5 UC attached separately.

Use case name	#2.5.a Browse for products by different options
Actors	User
Pre-conditions	1. The current situation is according to Use-Case 2.1
Post-conditions	All the products that are presented are compatible with the chosen option.
Input Parameters	Search options, search text
Main success scenario	1. **User:** Selects what to look for in the trading system and delivers the keywords according to the option. 2.1 **User**: If User selects: by product name a. **TradeSystem**: Checks if the text that was inserted is valid b. **TradeSystem**:Returns a list of products whose name is related to the search words 2.2 **User**: if User selects by product category: a. **TradeSystem**: Checks if the text that was inserted is valid b. **TradeSystem**: Returns a list of products that belong to the same category the user provided. 2.3 **User**: if User selects by product keywords: a. **TradeSystem**: Checks if the text that was inserted is valid b. **TradeSystem**: Returns a list of products that belong to the same category the user provided.
Alternatives/Extensions	~2.1.a returns an empty list ~2.2.a returns an empty list ~2.3.a returns an empty list
Trigger	User ask to browse for products related to different options.

Action	Data	Expected Result
#2.5.a (AT1) Browse for products by product name	Product name that exists in the system, proper and legal.	Success: Products that are related with their names for the user's input name.
	Empty input.	Failure: User should provide a non-empty keyword.
	Keywords that do not exist in the name of any product in the system.	Failure: returns empty set of products
#2.5.a (AT2) Browse for products by category	Product category that exists in the system, proper and legal.	Success: Products that are related with their category for the user's input.
	Empty input.	Failure: User should provide a non-empty keyword.
	Keywords that do not exist in the name of any product in the system.	Failure: returns empty set of products
#2.5.a (AT3) Browse for products by keywords	Product category that exists in the system, proper and legal.	Success: Products that are related with their keywords for the user's input.
	Keywords that do not exist in the name of any product in the system.	Failure: returns empty set of products

Action	Data	Expected Result
#2.5.a (AT4) Browse for products by Spelled wrong keywords	Inserts words with grammatical or letter errors	Success: find products according to similar words.
	Inserts incorrect, missing, or nonexistent keywords.	Failure: An error message is displayed according to the type of error (incorrect or missing details or keyword does not exist).

Use case name	#2.5.b Filter results according to characteristics
Actors	User
Pre-conditions	1. The current situation is according to Use-Case 2.5.a
Post-conditions	1. List of products according to the user's filtering characteristics
Input Parameters	(2.5.a) products list, filters list
Main success scenario	1. **User:** Chooses the option by which he wants to filter the results. 2.1 **User**: If User selects to filter by price range: a. **TradeSystem**: Checks if the input that was inserted is valid b. **TradeSystem**: Returns a list of products that are in the price range entered as input. 2.2 **User**: If User selects to filter by product rating: a. **TradeSystem**: Checks if the text that was inserted is valid b. **TradeSystem**: Returns a list of products that are in the rating entered as input and above. 2.3 **User**: If User selects to filter by category: a. **TradeSystem**: Checks if the text that was inserted is valid b. **TradeSystem**: Returns a list of products that are in the category entered as input. . . . (Could be more filters and combinations of them)
Alternatives/Extensions	~2.1.a returns an empty list. ~2.2.a returns an empty list. ~2.3.a returns an empty list.
Trigger	GuestUser ask to filter results (from2.5.a) by different characteristics.

Action	Data	Expected Result
#2.5.b (AT1) Filter results by Price Range	Price range and products list when at least some of the products in the list are within the range entered.	Success: Returns a product list that includes only products that are in range.
	Price range and product list when none of the products in the list are within the range entered.	Failure: Returns an empty list.
	Empty list	Failure: User should provide a non-empty list of products.
#2.5.b (AT2) Filter results by Rating	Rating and products list when at least some of the products have ratings like the input and above.	Success: Returns a product list that includes only products that are in the rating and above.
	Rating and products list when none of the products have ratings like the input and above.	Failure: Returns an empty list.
	Empty list	Failure: User should provide a non-empty list of products.
#2.5.b (AT3) Filter results by category	Category and products list when at least some of the products in the list are within the category.	Success: Returns a product list that includes only products that are in range.
	Category and products list when none of the products are in the same category.	Failure: Returns an empty list.
	Empty list	Failure: User should provide a non-empty list of products.

*There are also different combinations for the filters and all tests will look similar.

Use case name	#2.6 Saving products in User's basket for specific store
Actors	User
Pre-conditions	1. The current situation is according to Use-Case 2.5
Post-conditions	1. User has a basket with saved products.
Input Parameters	Products list.
Main success scenario	1. **User**: Choose product from products list to add to the basket 2. **TradeSystem**: returns product's page 3. **User**: if the User choose to add the product to the basket: a. **TradeSystem**: adds to the user's-store basket the chosen product b. Go back to step 1 (for more products)
Alternatives/Extensions	~3. Returns products list.
Trigger	GuestUser choose product from the products list.

Action	Data	Expected Result
#2.6 Saving products in User's basket for specific store	Selects and saves a product/s to a specific store's basket	Success: The product is stored in the user-store basket during the user's current visit.
	Select product that currently not available and tries to add it to the basket	Failure: The basket remains unchanged.
	No product is selected	Failure: User should choose at least one product.

Use case name	#2.7 View and edit shopping cart

Actors	User
Pre-conditions	The current situation is according to Use-Case 2.1.
Post-conditions	1. User's cart edited according to his actions
Input Parameters	None.
Main success scenario	1. **User**: The user wants to view the shopping cart 2. **TradeSystem**: returns the shopping cart of the user. 3 **User**: If User selects to edit the cart: a. **TradeSystem**: returns the edit options (delete, change quantity, and more) b. **User**: Choose products to edit, and how to edit them c. **TradeSystem**: The system makes the changes requested by the User.
Alternatives/Extensions	~3. Cart remains unchanged.
Trigger	User ask to edit his shopping cart

Action	Data	Expected Result
#2.7 - (AT1) Delete product from shopping cart	Products(>=1) from the cart.	Success: The selected products are deleted from the cart and no longer appear in it.
	No products selected.	Failure: User should choose at least one product.
#2.7 - (AT2) Change quantity product from shopping cart	Product from the cart.	Success: The quantity of the product selected in the cart was changed.
	No products selected.	Failure: User should choose some product.
	Products(>1) from the cart.	Failure: User should choose maximum one product.

*System sequence diagram of 2.8 UC attached separately.

Use case name	#2.8.a Purchase products according to the types of purchase
Actors	User
Pre-conditions	1. The current situation is according to Use-Case 2.1 2. User has a products list from Use-Case 2.5.
Post-conditions	1. The store received payment from the user for the product. 2. The user received the product for which he paid.
Input Parameters	Products list
Main success scenario	1. **User**: Asks to buy a product from the productsList 2. **User**: Choose : 2.1. If the User choose to buy a product in immediate purchase: a. **TradeSystem**: If the user can buy the product according to the store's purchase policy: 1. **TradeSystem:** If the product is sold immediately: a. **TradeSystem:** If the product is in stock: b. **TradeSystem:** Asks User's payments details and shipping address. c. **User**: Provides payment and shipping details. d. **TradeSystem:** Sends the payment details to the payment system. e. **Payment:** If the user have enough money f. **TradeSystem:** Sends the shipping details to the shipment system. **g. Shipment**:
Alternatives/Extensions	~2.1.a The buying process has stopped. ~2.1.a.1 The buying process has stopped.
Trigger	User ask to buy a product in immediate purchase.

Action	Data	Expected Result
#2.8.a (AT1) Immediate Purchase	A product that is sold in immediate purchase, is in stock and meets the store's purchase policy from which it is sold to the user. A user who meets the store policy conditions for buying the product and has enough money.	Success: The transaction took place, the inventory in the store was updated, the money went from the buyer to the store, the product went from the store to the buyer.
	A product that is sold in immediate purchase, is **not** in stock and meets the store's purchase policy from which it is sold to the user. A user who meets the store policy conditions for buying the product and has enough money.	Failure: The shopping process has stopped and continues as long as the product inventory in the store has not been updated.
	A product that is sold in immediate purchase, is in stock and does **not** meet store purchase policy conditions. A user who meets the store policy conditions for buying the product and has enough money.	Failure: The shopping process has stopped.
	A product that is sold in immediate purchase, is in stock and meets the store's purchase policy from which it is sold to the user. A user who does **not** meets the store policy conditions for buying the product and has enough money.	Failure: The shopping process has stopped.

Use case name	#2.8.b Purchase products according to discount types
Actors	User
Pre-conditions	1. The current situation is according to Use-Case 2.1 2. User has a products list from Use-Case 2.5.
Post-conditions	1. The store received payment from the user for the product. 2. The user received the product for which he paid.
Input Parameters	Products list
Main success scenario	1. **User**: Asks to buy a product from the products list 2. **User**: Choose : 2.1. If the User choose to buy a product with **visible discount**: b. **TradeSystem**: If the user can buy the product according to the store's purchase discount policies: 2. **TradeSystem:** If the product is sold with visible discount: h. **TradeSystem:** If the product is in stock i. **TradeSystem:** Asks User's payments details and shipping address. j. **User**: Provides payment and shipping details. k. **TradeSystem:** Sends the payment details to the payment system. l. **TradeSystem:** Sends the shipping details to the shipment system.
Alternatives/Extensions	~2.1.a The buying process has stopped. ~2.1.a.1 The buying process has stopped.
Trigger	User ask to buy a product with visible discount

Action	Data	Expected Result
#2.8.b (AT1) Visible Discount	A product that is sold at a visible discount, is in stock and meets the store's purchase policy from which it is sold to the user. A user who meets the store policy conditions for buying the product and has enough money.	Success: The transaction took place, the inventory in the store was updated, the money went from the buyer to the store, the product went from the store to the buyer.
	A product that is sold at a visible discount, is **not** in stock and meets the store's purchase policy from which it is sold to the user. A user who meets the store policy conditions for buying the product and has enough money.	Failure: The shopping process has stopped and continues as long as the product inventory in the store has not been updated.
	A product that is sold at a visible discount, is in stock and does **not** meet store purchase policy conditions. A user who meets the store policy conditions for buying the product and has enough money.	Failure: The shopping process has stopped.
	A product that is sold at a visible discount, is in stock and meets the store's purchase policy from which it is sold to the user. A user who does **not** meets the store policy conditions for buying the product and has enough money.	Failure: The shopping process has stopped.

Differences between GuestUser and RegisteredUser.

Use Case **2.4** performs the same for RegisteredUser.

Use Case **2.5** performs the same for RegisteredUser.

Use Case **2.6** performs differently for RegisteredUser: the Basket and the cart has been saved in his watch list for later date.

Use Case **2.7** performs the same for RegisteredUser.

Use Case **2.8** performs the same for RegisteredUser.

3.Registered User

Use case name	#3.1 Logout
Actors	RegisteredUser
Pre-conditions	Have valid registration, User state- Logged in.
Post-conditions	User state become GuestUser
Input Parameters	None.
Main success scenario	1. **TradeSystem** ask: “are you sure” If you press yes, the system logs out the **RegisteredUser**, the **RegisteredUser** becomes unlogged.
Alternatives/Extensions	User press cancel, the RegisteredUser stays logged in.
Trigger	The RegisteredUser wanted to logout.

Action	Data	Expected Result
#3.1 AT - Logout	None	User Logged out
	None	

Use case name	#3.2 Open store
Actors	RegisteredUser
Pre-conditions	1. Begin logged in. 2. Store name doesn't exist.
Post-conditions	1. New store opens to buy the RegisteredUser. 2. Registered RegisteredUser become the main StoreOwner.
Input Parameters	1. Store name. 2. Store purpose.
Main success scenario	1. **RegisteredUser:** enter new store name 2. **TradeSystem:** checks Store name doesn't exist. 2.1 **TradeSystem:** askes RegisteredUser's identifications . 2.2 **RegisteredUser:** enter the identifications. 3. New Store opens in the system 4. **RegisteredUser** becomes a StoreOwner in this store.
Alternatives/Extensions	1. **RegisteredUser:** enter same StoreName that exists already 1.1 **TradeSystem:** Display Error ("Store name already exists") 1.2 **TradeSystem** asks RegisteredUser for new store name.
Trigger	RegisteredUser want to open a store.

Action	Data	Expected Result

#3.2 AT - Open Store	Free to use store name	New Store opens with the chosen name
	User choose name that is already exists	Failure: getting an error message about the name.

Use case name	#3.7 Purchase history
Actors	RegisteredUser
Pre-conditions	None.
Post-conditions	None.
Input Parameters	RegisteredUserID, specific dates of purchase history
Main success scenario	1. **RegisteredUser** asks the system his purchase history. 2. **TradeSystem** displays the user's purchase history.
Alternatives/Extensions	1. **RegisteredUser** purchase history is empty.
Trigger	RegisteredUser asks to view his own purchase history.

Action	Data	Expected Result
#3.7 AT - Purchase history	Specific date range	Display Purchase history
	Wrong date – not real calendar date	Failure: error message about the fake date

4.Store Owner

Use case name	#4.1 StorageManaging
Actors	StoreOwner
Pre-conditions	None.
Post-conditions	Store Storage changes.
Input Parameters	1. StoreOwner
Main success scenario	1.**StoreOwner** Add new product to store. 2. **StoreOwner** Delete exists product from the store. 3.**StoreOwner** Edit product details.
Alternatives/Extensions	1.There are no items to delete. 2.The store is empty, no items to edit.

Action	Data	Expected Result
#4.1 AT - StorageManaging	Add item to store	Item successfully added
	Delete non exist item	Failure: error message

Use case name	#4.3 Add new StoreOwner.
Actors	StoreOwner

Pre-conditions	The StoreOwner should be StoreOwner in his Store.
Post-conditions	An exist RegistertUser become StoreOwner.
Input Parameters	RegisterdUserID of the new StoreOwner.
Main success scenario	1.**StoreOwner:** enter the user's details. 2.**RegisteredUser:** become StoreOwner.
Alternatives/Extensions	1.The User is already StoreOwner.
Trigger	StoreOwner asks to change an RegisteredUser to StoreOwner.

Action	Data	Expected Result
#4.3 AT – Add new StoreOwner	UserID to become StoreOwner	User is now the new StoreOwner
	UserID of someone who is already StoreOwner	Failure: an error message about invalid UserID

Use case name	#4.5 Add new StoreManager.
Actors	StoreOwner
Pre-conditions	The user should be StoreOwner in his Store.

Post-conditions	An exist RegisteredUser become StoreManager.
Input Parameters	RegisteredUserID of the new StoreManager.
Main success scenario	1.**StoreOwner** enter the RegisteredUser's details. 2.**RegisteredUser** become StoreManager.
Alternatives/Extensions	1.The User is already StoreManager
Trigger	StoreOwner asks to change an RegisteredUser to StoreManager.

Action	Data	Expected Result
#4.5 AT- add new store manager	UserID to become StoreManager	User is now the new StoreManager
	UserID of someone who is already StoreManager.	Failure: an error message about invalid UserID

Use case name	#4.6 Change StoreManager permissions.
Actors	StoreOwner
Pre-conditions	StoreManger exists.
Post-conditions	StoreManager permissions changed.
Input Parameters	StoreManagerId, New permissions to add, permissions to delete.

Main success scenario	1. **StoreOwner** Enter StoreManagerID. 1.1 **StoreOwner** Adds permissions 1.2 **StoreOwner** Deletes permissions.
Alternatives/Extensions	1. There is no StoreManager in the store. 1.1 Display Error(“No manager found”)
Trigger	StoreOwner asks to change an StoreManager permissions.

Action	Data	Expected Result
#4.6 AT - Change StoreManager permissions.	StoreManagerId, New permissions to add	New permissions added to the StoreManager.
	StoreManagerId ,permissions to that doesn’t exits.	Failure: an error message about invalid permission to delete.

Use case name	#4.7 Delete StoreManager
Actors	StoreOwner
Pre-conditions	At least 1 Store manager in the store.
Post-conditions	StoreManager become RegisteredUser.
Input Parameters	StoreManagerID.
Main success scenario	1. **StoreOwner** Enter StoreManagerID 1.1 **StoreOwner** deletes **RegisteredUser**.
Alternatives/Extensions	1. The StoreManager is not created by this StoreOwner

Trigger	StoreOwner asks to delete an StoreManger from his store.

Action	Data	Expected Result
#4.7 AT - Delete StoreManager	StoreManagerID	StoreManager deleted .
	StoreManagerID that not created by the actor.	Failure: an error message about invalid StoreManager to delete.

Use case name	#4.10 Watch Store purchase history
Actors	StoreOwner
Pre-conditions	None
Post-conditions	None
Input Parameters	None
Main success scenario	1. **TradeSystem**: show a list of all purchase history 2. **StoreOwner**: choose to see all history or part of it.
Alternatives/Extensions	1. **StoreOwner** have no purchase history. 2. **StoreOwner**: choose specific time to view 2.1 **TradeSystem** Display Error(“no purchase history”)

Trigger	StoreOwner asks to view the purchase history in his own store.

Action	Data	Expected Result
#4.10 AT - Watch Store purchase history	History of specific exist date	Display the purchase history of the store on the specific date
	History of specific non exists date	Failure: an error message about invalid Specific date to view.

5.Store Manager

Use case name	#5.1 Manage Store
Actors	StoreManager
Pre-conditions	1. Current state – StoreManager 2. StoreManager must be authorized to carry out management action
Post-conditions	None
Input Parameters	None
Main success scenario	1. **StoreManager** carries out management action. 2. Store is successfully updated
Alternatives/Extensions	~1. **StoreManager** is not authorized to carry out management action. Display Error (“Unauthorized action”).
Trigger	Appointed StoreManager attempts to update store

Action	Data	Expected Result
#5.1 Manage Store	StoreManager attempts to carry out management action he is authorized to do	Action is carried out
	StoreManager attempts to carry out management action he is not authorized to do	Failure: an error message about attempting unauthorized action..

6.Admin

Use case name	#6.4 View Purchase History
Actors	SystemManager
Pre-conditions	1. Current state - SystemManager
Post-conditions	1. Purchase history is displayed
Input Parameters	StoreID or Username
Main success scenario	1. **TradeSystem**: presents 2 options: (“provide store ID” or “provide Username”) 2.1 **SystemManager**: If **SystemManager** selects “provide store ID” a. **TradeSystem**: checks if the storeID exists b. **TradeSystem**: Displays store’s purchase history 2.2 **SystemManager**: if **SystemManager** selects “provide Username” a. **TradeSystem**: checks if the Username exists b. **TradeSystem**: Displays user’s purchase history
Alternatives/Extensions	~2.1.a. Display Error (“StoreID not found!”). ~2.2.a. Display Error (“Username not found!”).
Trigger	SystemManager looks up purchase history

Action	Data	Expected Result
#6.4 View Purchase History using Username	Valid Username	Display User's purchase history
	Invalid Username	Failure: an error message about invalid Username.
#6.4 View Purchase History using StoreID	Valid StoreID	Display store's purchase history
	Invalid StoreID	Failure: an error message about invalid StoreID.
#6.4 View Purchase History	Null	Failure: an error message about invalud input.

Use case name	#7 Payment Collection
Actors	PurchaseManager, PaymentCollection
Pre-conditions	1. Transaction details are valid
Post-conditions	1. Payment processed for exact amount written in transaction details
Input Parameters	Transaction details
Main success scenario	1. **PurchaseManager** transfers transactions to **PaymentCollection** 2. **PaymentCollection** charges user's account 3. **PaymentCollection** returns confirmation of payment.
Alternatives/Extensions	~2. Payment cannot be processed due to: ~ Transaction details are invalid Display Error("Payment cannot be processed")
Trigger	A Transaction has taken place, need to charge user

Action	Data	Expected Result
#7 Payment Collection	Valid Transaction details	Confirmation that package shipment request was received by Shipment
	Invalid Transaction details	Failure: an error message about invalid Transaction details.

Use case name	#8 Package Shipment
Actors	PurchaseManager, Shipment

Pre-conditions	1. Package details are valid 2. User details are valid
Post-conditions	none
Input Parameters	Package details, User details
Main success scenario	1. **PurchaseManager** transfers package and user details to **Shipment**. 2. **Shipment** verifies details. 3. **Shipment** returns confirmation that shipment request has been accepted.
Alternatives/Extensions	~2.:Display Error (“Shipment Error”) ~ User’s info is incorrect ~ Package details are insufficient
Trigger	A transaction has taken place need to ship package to user

Action	Data	Expected Result
#8 Package Shipment	Valid User and Package details	Confirmation that package shipment request was received by Shipment
	Invalid User details and valid package details	Failure: an error message about invalid User details.

	Valid User details and invalid package details	Failure: an error message about invalid Package details.