Index

# System Section1 ---------------------------------------2-7

# User Section 2 ----------------------------------------8-47

## Activities as guest visitor and register----------------------------8-13

## Guest Visitor Purchase Activities---------------------------------13-20

## Purchase Activities of a Subscribed Visitor in the Market----21-28

## Purchase Activities of a Subscribed Visitor in the Market----29-44

## Activities as a store manager--------------------------------------45-46

## Activities as a system manager-----------------------------------46-47

# System, section 1:

## Use case 1: Initialize Marketplace System.

1. **Actor:** User
2. **Trigger:** User requested to Initialize the system.
3. **Preconditions:**

* At least one System Administrator is already registered in the system.
* A valid payment service is available and connected.
* A valid supply service is available and connected.

1. **Parameters:** System Details (delivery service, payment service, user Info).
2. **Main Scenario:**

* The system boots up.
* The system checks the availability and validity of the connected payment service.
* The system checks the availability and validity of the connected supply service.
* The system verifies that at least one user is assigned the System Administrator role.
* The system validates that all integrity rules are upheld.
* The system is successfully initialized and ready for user interaction (visitors, sellers, and administrators).

1. **Alternative Flows:**

* **Payment service is not available**» System notifies that the payment service could not be reached or is invalid.
* **supply service is not available**  
  » System notifies that the supply service could not be reached or is invalid.
* **No system administrator found**  
  » System notifies that at least one system administrator must be defined.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| SuccessfulInitialization | 1. Valid payment service is available  2. Valid supply service is available  3. At least one system administrator exists | 1. Payment and supply services are added  2. System administrator is added  3. System is initialized and ready |
| MissingPaymentService | 1. supply service is available  2. No payment service is configured  3. System administrator exists | 1. System shows error for missing payment service  2. System is not initialized |
| MissingsupplyService | 1. Payment service is available  2. No supply service is configured  3. System administrator exists | 1. System shows error for missing supply service  2. System is not initialized |
| NoSystemAdmin | 1. Payment and delivery services are available  2. No system administrator is registered | 1. System shows error for missing system administrator  2. System is not initialized |
| IntegrityViolation | 1. All services and admin are set  2. System detects a rule violation | 1. System shows integrity violation error  2. System is not initialized |

## Use case 2: Payment Processing.

1. **Actor:**User
2. **Trigger:** User requests to process payment for a purchase.
3. **Precondition**:

* At least one recognized and available payment service is configured.
* User is registered and authenticated (logged in).
* User has at least one item selected for purchase.

1. **Parameters:** , Parameters: token, userId, shoppingCart, paymentDetails
2. **Main Scenario**:

* The user initiates a payment for the selected items.
* The system collects the transaction details (user info, amount, and selected payment method).
* The system selects a recognized and available payment service.
* The system sends the transaction details to the payment service.
* The payment service processes the transaction and returns a success confirmation.
* The system marks the payment as successful and continues the purchase process

1. **Alternative flow:**

* **Transaction details are invalid or incomplete**

» System detects missing or incorrect data and cancels the request.

* **No payment service configured**

» System shows a configuration error.

» Payment cannot be processed.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Result |
| Successful Payment | 1.User is logged in 2.Items selected 3.Valid payment service is available ,Valid payment details | System processes the payment successfully Transaction marked as paid |
| Payment Service Unavailable | 1.User is logged in 2.Items selected 3.Selected payment service is offline/unavailable | System shows an error message Payment is not processed |
| Missing Payment Details | 1.User is logged in 2.Items selected 3.Missing or incorrect transaction | System shows validation error Request is rejected |
| No Payment Service Configured | 1.User is logged in 2.Items selected 3.No payment service is set up in the system | System shows configuration error Payment cannot be completed |

## Use case 3 supply processing

* 1. **Actor:** User
  2. **Trigger:**User completes a purchase, A paid order is confirmed and ready for supply.
  3. **Precondition:**
  + At least one recognized and available supply service is configured.
  + User is registered and authenticated.
  + A confirmed purchase transaction exists.
  1. **Parameters:** shipment Details (including user info, User Shipping Address, Supplier info, and package info).
  2. **Main Scenario:**
  + **The market selects a suitable supply service based on availability and shipping method.**
  + **The market sends a supply request containing package and customer details.**
  + **The supply service acknowledges receipt of the request.**
  + **The supply service processes the shipment and returns a shipping confirmation.**
  + **The market updates the order status to "Shipped" and notifies the user.**
  1. **Alternative flow:**
* **Supply service unavailable**

» System shows an error and does not initiate supply.

* **Supply is rejected by the supply service**

» System notifies the user or admin of supply failure.

» Supply is not marked as confirmed.

* **Invalid supply details**

» System detects missing or incorrect customer/package info.

» System cancels the supply request and logs the error.

* **No supply service configured**

» System shows configuration error.

» Supply cannot be processed.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Result |
| Successful Supply | Valid supply service available Valid customer and package info | System sends request and receives confirmation Supply is marked as confirmed |
| Supply Service Down | Supply service is unavailable | System shows error Supply not processed |
| Supply Rejected | Supply service is available Service rejects the request | System shows rejection message Supply not confirmed |
| Invalid Supply Data | Missing customer or package info | System cancels request and shows error |
| No Supply Service Setup | No supply service configured in system | System shows configuration error Supply cannot be completed |

## Use case 4: Real-Time Notifications

* 1. **Actor:** System
  2. **Trigger:** A relevant event occurs that requires notifying the user in real-time.
  3. **Precondition:**
* The market system is operational
* Users are registered and logged into the system

**4.Parameters:** senderId, receiverId(s), message

5.**Main Scenario:**

* System detects an event that requires a notification.
* System generates a notification message.
* System sends the notification (or stores it if user is offline).
  1. **Alternative flow:**
* User try to purchase from closed store.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Result |
| Notification Sent (User Online) | Registered user is online  A relevant event occurs (e.g., message received) | Notification is delivered to the user in real time |
| Notification Saved (User Offline) | Registered user is offline  A relevant event occurs | Notification is saved  Delivered when user logs in |
| Notification Retry on Failure | User is online  Notification fails due to temporary error | System retries sending  If it fails again, notification is saved and logged |
| No Notification (User Not Registered) | Event occurs, but recipient is not a registered user | No notification is sent  System logs the issue |

## Use case 5: Delayed Notifications

* + 1. **Actor** :System
    2. **Trigger:**A relevant event occurs that requires notifying the user who is currently offline
    3. **Precondition:**
* The recipient is a registered subscriber.
* A notification event has occurred.
* The user is currently offline in the market.
  + 1. **Parameters:** userId
    2. **Main Scenario:**
* System detects an event that requires a notification.
* System generates a notification message based on the event type.
* System saves the notification.
* When the User login the system will send the notification

1. **Alternative flow**:

* **Notification Data is Invalid**

» Notification is not stored.

» System logs the issue for review.

* **User Never Logs Back In**

» Notification remains stored in the system until expiration or manual removal.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Result |
| Store Notification for Offline User | Registered user is offline  A notification is triggered | Notification is stored for future delivery |
| Deliver Stored Notification | User logs in after being offline  Stored notification exists | Notification is delivered and displayed to the user |
| Invalid Notification Data | Notification is missing user ID or message | Notification is not stored  Error is logged |
| User Never Logs In | Notification is saved  User does not return | Notification remains stored until expiration |

# Users, section 2:

## 1.Activities as guest visitor and register:

### Use case 1 - Enter as Guest

|  |  |
| --- | --- |
| Use Case Description | |
| Name | Enter as Guest |
| Actor | User |
| Trigger | Visiting the market and explore all the items and stores in it. |
| Precondition | 1 The system is properly initialized. |
| Parameters | There is no need for params. |
| Main Scenario | 1. The system boots up.  2.The user request to explore all markets.  3.the user receive number of markets and can load more until get all markets.  4. a temporary shopping cart created for the visitor and session token . |
| Alternative flow | 1.There are no markets in the system .  » System notifies that there are no market on the system.  2.internal error occurred with the server » System notifies a message to try again later. |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| SuccessfulExplore | 1. There are X number of markets | 1. User guest can see all of them  2.User guest can see all the items on each market  3. |
| NoMarkets | 1. no markets on the system | 1. System shows a message that there are no markets |
| NoItemsOnallMarkets | 1. markets that has no items | 1. System shows a message that there are no items on the all markets. |

### Use case 2 - Exit as Guest

|  |  |
| --- | --- |
| Use Case Description | |
| Name | Exit as Guest |
| Actor | User/Guest |
| Trigger | Exiting the market while not registered |
| Precondition | The market system is running and was explore on it . |
| Parameters | token |
| Main Scenario | 1. The system is running and available for users.  2.User guest entered the system and explore it.  3. user exits from the system.  4. the shopping cart associated with this guest will disappear. |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| SuccessfulExiting | 1.user without login or register | 1. guest exit successfully and the shopping cart deleted. |
| NoMarkets | 1. no markets on the system | 1. System shows a message that there are no markets |

### Use case 3 –Register

|  |  |
| --- | --- |
| Use Case Description | |
| Name | Register for a new user. |
| Actor | User/ Guest |
| Trigger | New user register to the system. |
| Precondition | 1 The system is properly initialized.(according to his usecase) |
| Parameters | token, username, password,age |
| Main Scenario | 1.the system is running and available.  2. The user guest open the system site.  3. user try to register and entered a valid info.  4. user registered successfully .  5. return to home page as a register. |
| Alternative flow | 1.username already exist in the system. |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Successfulregister | 1. user entered the system. | 1. user registered successfully.  2. return to home page as a register. |
| notValidParams | 1. invalid username or password. | 1. System shows a proper error massage. |

### Use case 4 – Login

|  |  |
| --- | --- |
| Use Case Description | |
| Name | Logging in to the system. |
| Actor | User guest |
| Trigger | User attempt to login. |
| Precondition | 1. The system is properly initialized  2. user already registered to the system.  3.User is logout. |
| Parameters | token, username, password |
| Main Scenario | 1.the user try to login using his username and password  2.the user logged in successfully.  3.created session token for this user . |
| Alternative flow | 1.there are no user with this username in the system.  2. wrong password.  3.the system shows a proper error massage according the problem that happen. |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results +token |
| SuccessfulLogin | 1. system is running .  2. user already registered to the system. | 1. User logged in successfully.  2.session token created. |
| Nonregistered | 1. None registered username | 1.The system shows a proper error massage |
| WrongPasswor | 1.Wrong password entered | 1.The system shows a proper error massage |

## 2.Guest Visitor Purchase Activities:

### Use case 1 - Explore markets

|  |  |
| --- | --- |
| Name | Explore markets |
| Actor | User |
| Trigger | User want to explore markets in the systems. |
| Precondition | 2. guest is in the system. |
| Parameters | token |
| Main Scenario | 1. The guest-visitor inputs the store id  2. The system shows the guest-visitor information about the requested store and its products.(\*\*) |
| Alternative flow | 1.There are no markets in the system .  2.System notifies that there are no market on the system. |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| SuccessfulExplore | 1. The guest enters the existing store that contain at least one product | 1. User guest can see all of them  2.User guest can see all the items on each market |
| NoMarkets | 1. the guest insert a non-existing store . | 1. System shows a message that there are no store in this id. |
| NoItemsOnMarket | 1. market that has no items . | 1. System shows a message that there are no items on the market. |

### Use case 2 - Search Filters

|  |  |
| --- | --- |
| Name | Search Filters. |
| Actor | User |
| Trigger | User requested to search a market with filter. |
| Precondition | 1.guest is in the system.  2.A valid supply service is available and connected and the store contain products. |
| Parameters | token, searchCriteria |
| Main Scenario | 1.User guest request to search market with filters.  3.User get a list of all markets that fit the filters .  4.User select a filters for items on the market.  5.User get a list of items that fit the filters. |
| Alternative flow | 1.There are no markets in the system .System notifies that there are no market on the system.  2.internal error occurred with the server System notifies a message to try again later. |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| SuccessfulSearch | 1.The guest-visitor entered correct search filiter for existing products. | 1. all the markets that fit the filters are on the list  2.there is no market that does not fit the filters. |
| NoMarketsFit | 1. no markets on the system that fit the filters  2.user without login or register | 1. System shows a message that there are no markets that fit the filters |
| SuccessfulItemFilter | 1.Market with items.  2.user without login or register | 1. all the markets that fit the filters are on the list  2.there is no market that does not fit the filters. |
| NoItemsOnMarket | 1. market that has no items  2.user without login or register | 1. System shows a message that there are no items on the market. |

### Use case 3 – add to Cart Case

|  |  |
| --- | --- |
| Actor | User Geust |
| Trigger | User requested to add an item to the cart. |
| Precondition | 1.guest is in the system.  3.A valid supply service is available and connected.(\*\*) |
| Parameters | token, itemToAdd |
| Main Scenario | 1.The user guest requests to explore a market in the system.  2.the user requests to add an item in the market to cart.  3.the item added to cart of user guest.  4.The user open the cart and find the item that he added . |
| Alternative flow | 1.There are no markets in the system .  System notifies that there are no market on the system.  2.Item is already added to cart  System will ask user to increase the quantity of item |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| SuccessfulAddToCart | 1. market contains items.  2.The guest-visitor enters a valid quantity for the product. | 1.The system updates the cart with the product. |
| Failureaddingtocart | 1. The guest-visitor enters the quantity of the product to be more than available in the store. | 1. The system notifies the user that the available quantity in the inventory is insufficient. |

### Use case 4 – Cart Edit

|  |  |
| --- | --- |
| Name | Explore markets |
| Actor | User Guest |
| Trigger | User requested to edit items on the cart. |
| Precondition | 1.user is in the system. |
| Parameters | token, itemToAdd, quantity |
| Main Scenario | 1.The user guest requests to explore the cart.  2.the user requests to add the quantity of one item.  3.the item quantity on the cart has been increased as user request. |
| Alternative flow | 1.the shopping cart is empty ,the system shows an empty cart.  . |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| SuccessfulEdit | 1. Cart with items. | 1. User guest can see all of the items on the cart after editing |
| FailureEdit | 1.add the amount of product that does not have more quantity | 1. System show notification that cant apply this edit . |

### Use case 5 – Buy the cart

|  |  |
| --- | --- |
| Name | Explore markets |
| Actor | User Guest |
| Trigger | User requested to buy the items on the cart. |
| Precondition | 1.User is in the system.  2.a cart with at least one item. |
| Parameters | token |
| Main Scenario | 1.Check policies for item and the user  2. availability of each product in cart  3. stock updates delete them from the stock of the store .  4. create receipt for the cart to the user .  5.delete the cart item and make it empty.  6.the system moves the user to payment process (service) with the cost of the cart.  7.the system create connection with the supply service.  8.User bought all the items on the cart.  9. the system notifies the user of the successful purchase. |
| Alternative flow | 1. not all the products are in the stock ,the system notifies the user without change the cart. |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| SuccessfulBuy | The guest buys the cart and the product are available | 1The order is completed successfully and moved to the supply service, and the user is notified of the success then clean the cart .(make it empty) |
| NoItem | The user try to buy the cart but find a product is insufficient | The order is not completed and the cart stay without changes |
| alternative | The connection with the payment service failed | System cancels order; cart remains unchanged; user notified. |

## 3.Purchase Activities of a Subscribed Visitor in the Market:

### Use Case 1: Logout

1. **Actor:** Subscribed Visitor
2. **Trigger:** The user requests to log out.
3. **Preconditions:**

* The user is logged into the system.
* The user is a registered member.

1. **Parameters:** SessionToken
2. **Main Scenario:**

* System verifies the session token.
* System marks the user as logged out.
* System stores the user's shopping cart for future visits.
* System redirects the user to the guest mode.

1. **Alternative Flows:**

* **Concurrent Logout from Multiple Devices :**

Only current device logged out; others remain active.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test name | Setup & Parameters | Expected Results |
| Successful Logout | 1. User 111 is created.  2. User 111 logs in.  3. User 111 initiates logout. | 1. Session is invalidated.  2. Shopping cart is saved.  3. User is redirected to guest mode. |
| Invalid Session Logout | 1.Invalid session token is used. | 1. System denies logout.  2. User receives an error message. |
| Partial Logout from One Device | 1.User 111 is created.  2. User 111 logs in on Device A.  3.User 111 logs in again on Device B.  4. User 111 initiates logout from Device A. | 1. Session on Device A is invalidated.  2. Session on Device B remains active.  3. User is redirected to guest mode on Device A.  4. User can still access system as logged-in user on Device B. |

### Use Case 2: Open Store

* 1. **Actor:** Subscribed Visitor
  2. **Trigger:** The user initiates a request to open a new store.
  3. **Preconditions:**
* The user is logged in.
* The user does not currently own the store requested.
  1. **Parameters:** token, storeName, category
  2. **Main Scenario:**
* System validates that the user can open a store.
* System registers the store with the given details.
* System assigns the user as the store’s founder.
* System confirms the store opening to the user.

1. **Alternative Flow – Open Store While Logged Out:**  
    User not logged in; store creation denied with login error.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test name | Setup & Parameters | Expected Results |
| Successful Store Open | 1. User 111 is created and logged in. 2. User 111 creates a store with name “\*\*\*\*”. | 1. Store with the name “\*\*\*\*” is registered and get id . 2. User 111 is assigned as owner. |
| Store Name already exists | 1. User 111 is created and logged in. 2. Store with the name “\*\*\*\*” already exists. 3. User 111 attempts to open a store with the same name. | 1. System rejects request with message error. |
| Open Store While Logged Out | 1. User 111 is created. 2. User 111 tries to open store “\*\*\*\*” without logging in. | 1. System denies the request. 2. User receives an error: “please log in to open store”. 3. Store “\*\*\*\*” is not created. |

### Use Case 4: Rate Product/Store

1. **Actor:** Subscribed Visitor
2. **Trigger:** The user requests to submit a rating.
3. **Preconditions:**

* The user has purchased the product/store being rated.
* The user is logged in.

1. **Parameters:** token, storeId, productId (for product), newRank
2. **Main Scenario:**

* System verifies that the user has purchased the product/store.
* System records the rating and review.
* System updates the average rating of the product/store.
* System confirms the submission.

1. **Alternative Flow – Multiple Ratings by Same User:**

New rating replaces old, average rating recalculated.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test name | Setup & Parameters | Expected Results |
| Valid Rating Submission | 1. User 111 purchases product A. 2. User 111 submits a 5-star rating. (the same for the store) | 1. Rating is stored. 2. Product A’s rating is updated |
| Unauthorized Rating | 1. User 222 has not purchased product A. 2. User 222 attempts to rate product A. | 1. Rating request is rejected |
| Multiple Ratings By Same User | 1. User 111 purchases (not mandatory purchase) product A. 2. User 111 rates product A 3 stars. 3. User 111 updates rating to 5 stars. | 1. Original rating is replaced. 2. Average product rating is recalculated. 3. Confirmation is shown to user. |

### Use Case 5: Send Messages to Store

1. **Actor:** Subscribed Visitor
2. **Trigger:** The user sends a message to a store.
3. **Preconditions:**

* The user is logged in.
* The store exists in the system.

1. **Parameters:** Parameters: senderId, receiverId, message
2. **Main Scenario:**

* System verifies the store's existence.
* The user write the message in input box.
* System sends the message to the store's message inbox.
* System notifies the store owner.
* System confirms the message submission to the user.

1. **Alternative Flow – Empty Message** 
   1. Empty message rejected; error shown; store not notified.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test name | Setup & Parameters | Expected Results |
| Successful Message Send | 1. User 111 logs in. 2. Store 222 exists. 3. User 111 sends to 222 a message “is this available?”. | 1. Message is stored. 2. Store owner is notified. |
| Store Not Found Message | 1. User 111 logs in. 2. Store 222 does not exist. 3. User sends a message to store 222. | 1. System return “store not found” error. |
| Empty Message Content | 1. User 111 is logs in. 2. Store 222 exists. 3. User 111 sends an empty message. | 1. System rejects the message. 2. User receives error:” message cannot be empty”. 3. Store owner is not notified. |

### Use Case 7: View Purchase History

1. **Actor:** Subscribed Visitor
2. **Trigger:** The user requests their purchase history.
3. **Preconditions:**

* The user is registered and logged in.
* The user has made past purchases.

1. **Parameters:** token
2. **Main Scenario:**

* System retrieves the user's past purchases.
* System presents the list of purchases.
* System ensures that removed products and stores remain in history.

1. **Alternative Flow – History Includes Removed Products:**
   1. Removed items shown with note; purchase history retained.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test name | Setup & Parameters | Expected Results |
| View History with Data | 1. User 111 is logged in. 2. User 111 has 3 completed purchases. | 1. System displays all 3 purchases. |
| View History No Data | 1. New user 222 with no purchases. | 1. System show “not purchases history found” or “the history is empty”. |
| History Includes Removed Products | 1. User 111 is logged in. 2. User 111 completes purchases of product A. 3. Product A has been removed from the store. 4. User 111 views purchases history | 1. Product A appears in the list. 2. System shows a note:” product A no longer available”. 3. Purchases data retained. |

### Use Case 9: Submit Purchase Bid

1. **Actor:** Subscribed Visitor
2. **Trigger:** The user submits a bid for a product.
3. **Preconditions:**

* The user is logged in.
* The product is available for bidding.

1. **Parameters:** token, bid
2. **Main Scenario:**

* System verifies that the product is available for bidding.
* System checks that the bid amount is valid (positive and greater than 0).
* System records the user's bid.
* System notifies the store owner in real-time about the new bid.
* System confirms the bid submission to the user according to the store owner comments(yes /no/ …).

1. **Alternative Flow – Invalid Bid Amount:**  
    Invalid bid denied; error shown; bid not stored.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test name | Setup & Parameters | Expected Results |
| Valid Bid Submission | 1. User 111 is logged in. 2. Product A allows bidding 3. User 111 places a bid of 50$. | 1. Bid is stored. 2. Store owner receives a real-time notification. 3. The store owner accept 4. The user add this product to his cart. |
| Closed Bid attempt | 1. User 111 is logged in. 2. Product A has closed the bidding. 3. User 111 attempts a bid. | 1. System rejects the bid. 2. User receives message: “Bidding is closed.” |
| Invalid Bid Amount | 1. User 111 is logged in. 2. Product A is open for bidding. 3. User 111 submits a bid lower than the current bid. | 1. System denies the bid. 2. User receives error:” invalid bid amount”. 3. No bid is stored. |

### Use Case 10: Buy Product in Auction

1. **Actor:** Subscribed Visitor
2. **Trigger:** The user wins an auction and proceeds with the purchase.
3. **Preconditions:**

* The auction has ended.
* The user is the highest bidder.

1. **Parameters:** token,AuctionID, PaymentDetails
2. **Main Scenario:**

* System verifies the auction outcome.
* System processes the payment.
* The user try to buy the product using confirm price.
* System confirms the purchase to highest bidder.

1. **Alternative Flow – Auction Loser Attempts Purchase:**
   1. Non-winner blocked from purchase; shown not winning message.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test name | Setup & Parameters | Expected Results |
| Auction Winner Checkout | 1. User 111 won the auction. 2. User submits payment. | 1. Payment processed. 2. Confirmation sent. 3. Add the won product to the cart of user 111 |
| Auction Payment Failure | 1. Payment method fails because of invalid payment details. | 1. Purchase canceled 2. User notified. |
| AuctionLoserAttemptsPurchase | 1. Auction for product A ends. 2. User 111 was not the highest bidder 3. User 111 attempts to purchase. | 1. User receives message:” you are not the winning bidder”. 2. No payment is processed. |

### Use Case 11: Buy Product in Lottery

1. **Actor:** Subscribed Visitor
2. **Trigger:** The user wins a lottery and proceeds with the purchase.
3. **Preconditions:**

* The lottery has ended.
* The user was randomly selected as the winner.

1. **Parameters:** LotteryID, PaymentDetails
2. **Main Scenario:**

* System verifies the lottery outcome.
* System processes the payment.
* System confirms the purchase.

1. **Alternative Flow – Lottery Loser Attempts Purchase:**
   1. Non-winner cannot buy; system shows rejection message.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test name | Setup & Parameters | Expected Results |
| Lottery Winner Purchase | 1. User 111 buy lottery for the product . 2. User 111 wins lottery for product A. | 1. Add the prouduct to the cart |
| Lottery Payment Error | 1. User’s payment fails because of invalid payment details. | 1. System cancels transaction. 2. Product not delivered. |
| Auction Loser Attempts Purchase | 1. Lottery ends. 2. User 111 did not win. 3. User 111 attempts purchase. | 1. User receives message: “you are not the selected winner”. 2. Product remains unpurchased by user. |

## 4.Visitor-Member as a store owner:

### Use-case: 1.a Stock management- AddProduct.

1. **Actor:** store owner.
2. **Trigger:** add a product to a specific store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.
* Product should be given.

1. **Parameters:** storeId, token, productId, quantity, price, category
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the User is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System adds the product successfully.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have permission.
* **No product provided:** System notifies the owner it cannot add nothing.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Product added successfully. | 1. Store A exists  2. User 111 is logged and he is store owner.  3. user attempt to add product X to store A | System validates the Store A exists  System validates user 111 is the store owner  System adds product X successfully  Product X appears in the store A stock |
| Store not found | 1. user 111 is logged in (store owner  2.user 111 attempt to add product X to not exist store | System notify the user that the store not found . |
| Member is not the owner | 1. Store exists  2. User is logged in but is not the store owner | System notifies the user it does not have permission. |
| No Product Provided | 1. Store exists  2. User is logged in as store owner  3. No product is given to add | System notifies the owner it cannot add nothing. |

### Use-case: 1.b Stock management- DeleteProduct.

1. **Actor:** store owner.
2. **Trigger:** delete a product from a specific store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.
* Product exists.

1. **Parameters:** storeId, token, productId
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the User is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System validates that the product exists.
* System deletes the product successfully.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have permission.
* **Product not found:** System notifies the owner the product cannot be deleted.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Product deleted successfully. | 1. valid Store exists  2. valid store owner exist and logged in  3. Valid product exists in the store  4.store owner request to deletion | System valid store ,ownership and product  Product is deletion successfully  Success notification sent |
| Store not found | 1. invalid store (not exist)  2. valid store owner logged in  3.delettion request. | System retuen notification the store not found  Not product is deleted. |
| Member is not the owner | 1. Store exists  2. User is logged in but is not the store owner | System notifies the user it does not have permission. |
| No Product Provided | 1. Store exists  2. User is logged in as store owner  3. No product is given to delete | System notifies the owner it cannot be deleted. |

### Use-case: 1.c Stock management- ChangeProduct.

1. **Actor:** store owner.
2. **Trigger:** change a product in a specific store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.
* Product exsits.

1. **Parameters:** storeId, token, productId, newQuantity, newPrice
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the User is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System validates that the product exists by the productId.
* System replaces the old product with the new one that is given.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have the permission.
* **Product not found:** System notifies the owner it cannot change it.
* **Not given product to change:** System notifies the owner it cannot change it with no new one to.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Product changed successfully. | 1. valid Store exists  2. valid User is logged in as store owner  3. Product exists by ID in the store  4. New valid description is given. | System validates store, ownership and product.  System update the changes . |
| Store not found | 1.invalid store that not exist  2. owner tries to change a product | System notifies the owner that the store is not found  No chages in the product in the store |
| Member is not the owner | 1. Store exists  2. User is logged in but is not the store owner | System notifies the user it does not have permission. |
| Product not found | 1. Store exists  2. User is logged in as store owner  3 Product ID is invalid or missing | System notifies the owner it cannot be changed. |

### Use-case:2.a Add purchase/sale policies in the store:

1. **Actor:** store owner.
2. **Trigger:** add new purchase or sale policies to the store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.
* New policy should be given.

1. **Parameters:** storeId, token, policyDetails
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the User is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System adds the new policy to the store.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have permission.
* **No policy provided:** System notifies the owner it cannot add nothing.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Policy added successfully. | 1. Store exists  2. User is logged in as store owner  3. Valid Policy is provided | Policy added successfully. |
| Store not found | 1. Store does not exist  2. User tries to add a policy | System notifies the user the store not found |
| Member is not the owner | 1. Store exists  2. User is logged in but is not the store owner | System notifies the user it does not have permission. |
| No Policy Provided | 1. Store exists  2. User is logged in as store owner  3. No policy is given to add | System notifies the owner it cannot add nothing. |

### Use-case: 2.b Delete purchase/sale policies in the store:

1. **Actor:** store owner.
2. **Trigger:** delete purchase or sale policies to the store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.
* policy should be given to delete.

1. **Parameters:** storeId, token, policyId.
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the User is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System delete the policy from the store.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have permission.
* **Policy not found:** System notifies the owner the policy cannot be deleted.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Policy deleted successfully. | 1. Store exists  2. User is logged in as store owner  3. Policy found in the store | Policy deleted successfully. |
| Store not found | 1. Store does not exist  2. User tries to delete a policy | System notifies the user the store not found. |
| Member is not the owner | 1. Store exists  2. User is logged in but is not the store owner | System notifies the user it does not have permission. |
| Policy not found | 1. Store exists  2. User is logged in as store owner  3. Policy not found int this store | System notifies the owner the policy cannot be deleted. |

### Use-case: 3 Add Ownership to a store.

1. **Actor:** store owner, new store owner.
2. **Trigger:** adding a new owner to the store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* “New store-owner” exists and identifies as a visitor-Member.
* Store-owner is logged in.

1. **Parameters:** storeId, token, newOwnerId
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the owner is a Visitor-Member and identifies as a store-owner.
* System validates that the newOwner is a Visitor-Member and not identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System makes an offer to the newOwner to give it an ownership.
* newOwner agrees and the System makes it an owner.

1. **Alternative flows:**

* **Member is not the owner:** System notifies the user it does not have the permission.
* **Member is already an owner:** System notifies the owner it not make the newOwner an Owner due to previous ownership.
* **newOwner denies the ownership:** System notifies the original owner that it cannot make the nominee owner an ownership due to rejection issue.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Ownership added successfully. | 1. Store exists  2. Owner is a logged-in and sent request to register user to add him as a owner  3. the register user accept the request | System makes the register user an owner of the store. |
| user is not the store owner | 1. Store exists  2. Owner is a logged-in and sent request to register user to add him as a owner  3. the register user already the owner | System notifies that the newOwner is already an owner and cannot be added. |
| newOwner denies the ownership | 1.Store exists  2. User is logged-in store owner  3.newOwner is a Visitor-Member 4. newOwner rejects the offer | System notifies the original owner that it cannot make the nominee owner an ownership due to rejection issue. |

### Use-case: 4 Delete Ownership from a store.

1. **Actor:** firststore owner, second store owner.
2. **Trigger:** deleting an owner from the store.
3. **Precondition:**

* Store exists.
* The two store owners exist and identify as a store owner.
* First store-owner is logged in.
* Store owner got his ownership from the first store owne”.

1. **Parameters:** storeId, token, ownerToDelete
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the two owners are a Visitor-Member and identify as a store-owner.
* System validates that these two owners own the given storeId.
* System validates that the second owner got his ownership from the first owner.
* System deletes the ownership from the second owner.
* System deletes all members who received their membership from the second owner.
* System deletes all managers who received their management from the second owner.

1. **Alternative flows:**

* **Store not found:** System notify the owner the store not found.
* **Member is not the owner:** System notify the user it does not have the permission to delete an ownership.
* **First Owner is not the one who gave the ownership:** System notify the owner it does not have the permission to delete an ownership.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Ownership deleted successfully. | 1. Store exists  2. First owner is a logged in and is the store-owner  3. Second owner is also the owner  4. the first owner must be the owner that add the second as owner | 1. System deletes the ownership from SecondOwner  2. Deletes all roles(worker who add them ) granted by SecondOwner |
| Store not found | 1. Store exists  2. First owner is a logged in and is the store-owner  3. Second owner is also the owner  4. the first owner is not the owner that add second owner | System notifies you can delete this owner don’t have the permission |
| Member is not the store owner | 1.Store exists  FirstOwner is logged in but is not a store owner  Or SecondOwner is not a store owner | System notifies the user that they do not have permission to delete ownership |
| FirstOwner Did Not Grant Ownership | 1. Store exists  2. FirstOwner is logged in and is a store owner  3. SecondOwner is also a store owner but was not appointed by FirstOwner | System notifies the user that they do not have permission to delete this ownership |

### Use-case:6 Add a manager to a store.

1. **Actor:** store owner, new manager.
2. **Trigger:** adding a manager to the store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner..
* Store-owner is logged in.
* Manager exists and a is not a store owner or a manager either.

1. **Parameters:** storeId, token, managerId
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the owner is a Visitor-Member and identifies as a store-owner.
* System validates that the new manager is a Visitor-Member and is not a manager already or an owner either.
* System validates that the owner owns the given storeId.
* System makes an offer to the new manager to give it management.
* New Manager agrees and the System makes it the Manager.
* Owner makes specific authorizations to the new manager.

1. **Alternative flows:**

* **Member is not the owner:** System notifies the user it does not have the permission to deal with management issues.
* **Member is already an owner/manager:** System notifies the owner it cannot make the new manager a manager due to their existing role.
* **Nominee manager denies the management deal:** System notifies the owner that it cannot give the nominee manager the management due to rejection issue.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Manager added successfully. | 1. Store exists  2. Store owner is logged in  3. store owner send request to register user to be manger of the store  4. the reg user accept | 1. System adds the new manager to the store  2. System grants the new manager specific authorizations |
| Member is not the store owner | 1.Store exists  2. User logged in is not the store owner | System notifies the user that they do not have permission |
| Member is already an owner/manager | 1. Store exists  2. New manager is already a store owner or manager | System notifies the owner they cannot make the user a manager due to their existing role |
| Nominee manager denies the management deal | |  | | --- | |  |  |  | | --- | | 1. Store exists  2. Nominee manager is offered the role 3. Nominee manager declines | | System notifies the owner that it cannot give the nominee manager the management due to rejection issue. |

### Use-case:7.a Add authorization for a specific manager.

1. **Actor:** store owner, manager.
2. **Trigger:** change the authorization for a manger in the store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.
* Manager exists.
* Manager got his management deal from this store owner.

1. **Parameters:** token, managerId, storeId, authorizations
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the owner is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System validates that the manager got his management deal form this owner.
* Owner adds newAuthorization this manager.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have the permission to deal with management issues.
* **Owner is not the one who made the management deal:** System notifies the owner it does not have the permission to deal with this management issue.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Authorization added successfully. | 1. Store exists  2. Store owner is logged in  3. Manager exists  4. Manager got their management deal from the store owner  5. New authorization is provided for the manager | System add the new authorization to the manager |
| Store not found | 1. Store does not exist  2. User tries to add authorization to manager in a non-existing store | System notifies the user that the store is not found |
| Member is not the store owner | 1.Store exists  2. User logged in is not the store owner | System notifies the user that they do not have permission |
| Owner is not the one who made the management deal | 1. Store exists  2. User is logged in as a store owner but is not the one who made the manager's management deal | System notifies the owner it does not have the permission to deal with this management issue. |

### Use-case:7.b Delete(change) authorization from a specific manager.

1. **Actor:** store owner, manager.
2. **Trigger:** change the authorization for a manger in the store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.
* Manager exists.
* Manager got his management deal from this store owner.

1. **Parameters:** token, managerId, storeId, authorizations
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the owner is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System validates that the manager got his management deal form this owner.
* Owner deletes authToDelete from this manager.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have the permission to deal with management issues.
* **Owner is not the one who made the management deal:** System notifies the owner it does not have the permission to deal with this management issue.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Authorization deleted successfully. | 1. Store exists  2. Store owner is logged in  3. Manager exists  4. Manager got their management deal from the store owner | System removes or changes the specified authorizations for the manager |
| Store not found | 1. Store does not exist  2. User tries to remove authorization from manager in a non-existing store | System notifies the user that the store is not found |
| Member is not the store owner | 1.Store exists  2. User logged in is not the store owner | System notifies the user that they do not have permission |
| Owner is not the one who made the management deal | 1. Store exists  2. User is logged in as a store owner but is not the one who made the manager's management deal | System notifies the owner it does not have the permission to deal with this management issue. |

### Use-case:9 Deactivate store.

1. **Actor:** store owner.
2. **Trigger:** deactivatea store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.

1. **Parameters:** storeId, token
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the User is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System deactivates the store.
* System notifies all the managers and the members of this store.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have the permission to close this store.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Store closed successfully. | 1. Store exists  2. Store owner is logged in  3. Store owner owns the store  Store owner send request to system to close his store | 1. System deactivates the store  2. System notifies all managers and members |
| Store not found | 1. Store does not exist  2. User tries to close the store | System notifies the user that the store is not found |
| Member is not the store owner | 1.Store exists  2. User logged in  3. User is not the store owner | System notifies the user that they do not have permission |

### Use-case:11 Get Information about store.

1. **Actor:** store owner.
2. **Trigger:** display information from the stores owned by the owner and view the authorizations granted to the management team.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.

1. **Parameters:** token, storeId
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the User is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System displays the information requested by the owner.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have the permission see this info.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Information displayed successfully. | 1. Store exists  2. Store owner is logged in  3. Store owner owns the store  4.store owner request display information of management team | System displays the store information, including authorizations for managers |
| Store not found | 1. Store does not exist  2. User requests info from the store | System notifies the user that the store is not found |
| Member is not the store owner | 1.Store exists  2. User logged in  3. User is not the store owner | System notifies the user that they do not have permission |

### Use-case:12 Messages in the store.

1. **Actor:** store owner, member.
2. **Trigger:** members of the store asking questions and the owner replies.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.
* Member exists.

1. **Parameters:** senderId, receiverId, message
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the User is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System validates that the user is validating as guest or register
* Owner replies to the message that he got from the member.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have the permission see this info.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Message replied successfully. | 1. Store exists  2. Store owner is logged in  3. Store owner owns the store  4. Sender as guest or register  5. guest or register sends a message | System allows the store owner to reply to the message |
| Store not found | 1. Store does not exist  2. Member tries to send a message | System notifies the user that the store is not found |
| User is not the store owner | 1.Store exists  2. User logged in  3. User is not the store owner tries to reply to a message | System notifies the user that they do not have permission |

### Use-case: 13 Show purchases history in the store.

1. **Actor:** store owner.
2. **Trigger:** showing the history of the purchases in the store.
3. **Precondition:**

* Store exists.
* Visitor-Member exists and identifies as a store owner.
* Store-owner is logged in.

1. **Parameters:** storeID.
2. **Main Scenario:**

* System searches and validates that the store exists.
* System validates that the User is a Visitor-Member and identifies as a store-owner.
* System validates that the owner owns the given storeId.
* System shows all the purchases history to the owner.

1. **Alternative flows:**

* **Store not found:** System notifies the owner the store not found.
* **Member is not the owner:** System notifies the user it does not have the permission see this info.

**Acceptance tests:**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| Purchase history displayed successfully. | 1. Store exists  2. Store owner is logged in  3. Store owner owns the store  4.store owner request to show history purchase of specific store by id | System displays all purchase history related to the store that provide the id |
| Store not found | 1. Store does not exist  2. User requests purchase history | System notifies the user that the store is not found |
| User is not the store owner | 1.Store exists  2. User logged in  3. User is not the store owner | System notifies the user that they do not have permission |

## 5.Activities as a store manager:

### Use case 1 – Store manager permissions

|  |  |
| --- | --- |
| Use Case Description | |
| Name | Store manager permissions |
| Actor | User Store |
| Trigger | Manager wants to do some management stuff. |
| Precondition | 1.User is logged in.  2.the user is a store manager.  5.the user has at least one permission . |
| Parameters | User as a store manager. |
| Main Scenario | 1.The store user add for the manager a permission to delete an item.  2.The store manager can delete an item. |
| Alternative flow | 1.There user is not logged in » System notifies that the user has no permission to delete items.  2.internal error occurred with the server » System notifies a message to try again later. |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| SuccessfulPermision | 1.user as a store manager and try to make manage in the store | 1. the manage is finish successfully |
| NoPermisions | 1. user as a store manager .  2.store manager has no permissions | 1. user cant apply the action |
| UserRemove | 1. user that is not store manager . | 1. User cant remove item. |

## 6.Activities as a system manager:

### Use case 1 – Close a store

|  |  |
| --- | --- |
| Use Case Description | |
| Name | Close a store |
| Actor | System manager |
| Trigger | System manager want to close a store. |
| Precondition | Store exist in the system. |
| Parameters | storeId, token |
| Main Scenario | 1.system manager enters the system.  2. manager chooses a store to close permanently.  3. store closes successfully.  4. all managers and owners for this store were notified .  5. all managers and owners for this store were no longer in this position. |
| Alternative flow | 1.There are no store with the given id in the system. |

**Acceptance tests**

|  |  |  |
| --- | --- | --- |
| Test Name | Setup & Parameters | Expected Results |
| SuccessfulStoreClose | 1. store exist in the system.  2. manager logged in | 1. store closed successfully.  2. all owners and managers were notified.  3. all owners and managers are no longer in this position. |
| NoStoreIdExist | 1. none existing store id entered. | 1. System shows a proper massage. |
| ManagerNotLoggedin | 1. Manager not logged in. | 1. System shows a proper massage. |