I. System Use-Cases:

1. Use-Case: Initialize Marketplace System

    1. Actor: System Admin
    2. Preconditions: None
    3. Parameters: Admin Credentials
    4. Postconditions:
        1. 'System Admin' is initialized
        2. System has established connections with external services (payment, delivery)
    5. Result: Marketplace system is initialized and ready for use
    6. Actions:
        1. System Admin: Runs the marketplace process
        2. System: Registers a 'System Admin' member
        3. System: Associates System Admin with 'System Admin' instance
        4. System: Establishes connections with payment and delivery services

2. Use-Case: Add connection with an external service

    1. Actor: System Admin
    2. Preconditions:
        1. Current marketplace state
        2. An active connection with another similar external service does not exist
    3. Parameters:
        1. External Service type
        2. All required parameters to establish connection with the service
    4. Postconditions:
        1. Current marketplace state (i.e. state has not been altered)
        2. An active connection with the external service exists
    5. Result: All traffic related to the external service is routed to it
    6. Actions:
        1. System Admin: Specifies external service to connect to
        2. System Admin: Specifies relevant details to allow connection
        3. System: Establishes connection with appropriate external service

2.1. Use-Case: Edit connection with an external service

1. Actor: System Admin
2. Preconditions:
    1. Current marketplace state
    2. An active connection with the external service of this type exists
    3. User is a system admin
3. Parameters:
    1. Session ID
    2. Username
    3. External Service Type
    4. Modification settings
4. Postconditions:
    1. Current marketplace state (i.e. state has not been altered)
    2. The connection with the external service has updated parameters
5. Result: All traffic related to the external service is routed to it according to the parameters specified
6. Actions:
    1. System Admin: Specifies modification details
    2. System: Forwards request to external service
    3. System: Updates external service state according to response

2.2. Use-Case: Swap connection with an external service

1. Actor: System Admin
2. Preconditions:
    1. Current marketplace state
    2. An active connection with an external service of this type exists
    3. User is a system admin
3. Parameters:
    1. Username
    2. External Service Type
    3. Modification settings
    4. External Service type
    5. All required parameters to establish connection with the service
4. Postconditions:
    1. Current marketplace state (i.e. state has not been altered)
    2. An active connection with the original external service does not exist
    3. An active connection with the new external service exists
5. Result: All traffic related to the external service is routed to the new service
6. Actions:
    1. System Admin: Specifies external service to connect to
    2. System Admin: Specifies relevant details to allow connection
    3. System: Disconnects from original external service
    4. System: Establishes connection with appropriate external service

3. Use-Case: Call Payment Service

1. Actor: System
2. Preconditions:
    1. A checkout operation has been performed by a user
    2. A connection with a payment service exists
3. Parameters: Order details (contains information regarding a specific transaction)
4. Postconditions: User's checkout has succeeded or failed
5. Result: Payment confirmation/refusal
6. Actions:
    1. System: Forwards order details to external service
    2. System: Receives external service response
    3. System: Returns response

4. Use-Case: Call Delivery Service

1. Actor: System
2. Preconditions:
    1. A checkout operation has been performed by a user
    2. A payment service has confirmed the transaction
    3. A connection with a delivery service exists
3. Parameters:
    1. Delivery details
    2. Client credentials
4. Postconditions: None
5. Result: Delivery request confirmation/refusal
6. Actions:
    1. System: Forwards order details to external service
    2. System: Receives external service response
    3. System: Returns response

5. Use-Case: Real-Time Notifications

1. Actor: System
2. Preconditions: Users are logged in
1. Parameters:
    1. Usernames
    2. Condition/Message
2. Postconditions: All users related to the satisfied conditions have a pending message
3. Result: None
4. Actions:
    1. System: Creates a message according to the satisfied condition
    2. System: Notifies all usernames a message is pending

6. Use-Case: Delayed Notifications

1. Actor: System
2. Preconditions: Users are logged out
3. Parameters:
    1. Usernames
    2. Condition/Message
4. Postconditions: Database contains messages destined for the specified users
5. Result: None
6. Actions:
    1. System: Creates a message according to the satisfied condition
    2. System: Stores all messages and their recipients' usernames

7. Use-Case: Notifications

1. Actor: System
2. Preconditions: One of the following conditions has been satisfied:
    - A client has purchased a product from a shop
    - A shop is closed
    - A shop is re-opened
    - A user nomination has been rescinded
    - A user received a message/inquiry
3. Parameters:
    1. Usernames
    2. Condition/Message
4. Postconditions: All users related to the satisfied conditions have a pending message
5. Result: None
6. Actions:
    1. System: Creates a message according to the satisfied condition
    2. System: Calls Real-Time Notifications for logged in members, calls Delayed Notifications for logged out members

II. User Related Use-Cases:

Guest Use-Cases:

1. General Guest Use-Cases:

1. Use-Case: Access Marketplace

    1. Actor: User
    2. Preconditions: Session ID for instance was created by the server
    3. Parameters: Session ID
    4. Postconditions:
        1. ‘Guest’ instance representing the user exists
        2. ‘Guest’ instance has an empty shopping cart
        3. ‘Guest' instance is associated with the user
    5. Result: User can perform general and purchase related actions
    6. Actions:
        1. System: Creates a new ‘Guest’ instance with an empty shopping cart
        2. System: Presents to the user relevant guest actions and data

2. Use-Case: Exit Marketplace (Guest)

    1. Actor: User
    2. Preconditions: User has an existing active profile
    3. Parameters: Session ID
    4. Postconditions:
        1. ‘Shopping Cart’ is emptied
        2. ‘Guest’ instance is deleted
    5. Result: User can no longer perform any actions within the system
    6. Actions:
        1. System: Empties the ‘Shopping Cart’
        2. System: Deletes the associated ‘Guest’ instance
        3. System: Closes marketplace system instance

3. Use-Case: Register

1. Actor: User
2. Preconditions:
    1. ‘Guest’ instance associated with the user exists
    2. A ‘Member’ with the same username does not exist in the system
3. Parameters:
    1. Session ID
    2. Username
    3. Password
4. Postconditions:
    1. New ‘Member’ instance exists
    2. The new ‘Member’ instance holds all identifying details given by the user
5. Result: A new ‘Member’ is added to the system
6. Actions:
    1. User: Inputs all relevant identifying details
    2. User: Confirms input
    3. System: Checks for data validity
        i. System: Finds that data is invalid
            a. System: Present error message
        ii. System: Finds that data is valid
            a. System: Create new ‘Member’ instance with the given identifying details

4. Use-Case: Login

1. Actor: User
2. Preconditions: ‘Guest’ instance associated with the user exists (the user is not logged in)
3. Parameters:
    1. Session ID
    2. Username
    3. Password
4. Postconditions:
    1. User is identified as ‘Member’ with its associated details
    2. User is associated with his unique shopping cart
5. Result: User can perform any member related operations
6. Actions:
    1. User: Inputs username
    2. User: Inputs password
    3. User: Confirms input
    4. System: Checks for data validity
        i. System: Finds that data is invalid
            a. System: Presents error message
        ii. System: Finds that data is valid
            a. System: Associate user with appropriate ‘Member’ instance

2. Guest Payment Use-Cases:

1. Use-Case: Get Shop Info

    1. Actor: User
    2. Preconditions: User has an associated 'User' instance (e.g. 'Guest' or 'Member')
    3. Parameters:
        1. Session ID
        2. Shop ID
    4. Postconditions: None
    5. Result: Display relevant shop info, including products that the shop is offering
    6. Actions:
        1. User: Requests shop details
        2. System: Verifies user has accessed the marketplace (guest) or is logged in (member)
        3. System: Searches for shop
        4. System: Finds that shop exists
            i. System: Displays relevant shop info
        5. System: Finds that shop doesn't exist
            i. System: Displays to user that shop wasn't found

2. Use-Case: Search Products

    1. Actor: User
    2. Preconditions: User has an associated 'User' instance
    3. Parameters:
        1. Session ID
        2. Keywords and filters
    4. Postconditions: None
    5. Result: Products corresponding to the given parameters
    6. Actions:
        1. System: Initialize search process
        2. User: Inputs keywords
        3. User: Inputs filters (Optional)
        4. User: Confirms input
        5. System: Searches according to the given parameters
        6. System: Displays the relevant products (or nothing if no products were found)

4.1. Use-Case: Add to shopping cart

1. Actor: User
2. Preconditions:
    1. User has an existing instance
    2. The user is the owner of the shopping cart
    3. A shop with the shop ID exists
    4. A product with the product ID exists in the relevant shop
    5. Desired product quantity is within the shop's stock
3. Parameters:
    1. Session ID
    2. Product ID
    3. Product Quantity
4. Postconditions: User's shopping cart contains the corresponding product
5. Result: None
6. Actions:
    1. User: Selects product to add
    2. System: Checks if product with product ID exists
    3. System: Checks that the product quantity does not exceed the shop's stock
    4. System: Adds product ID to the relevant shop's 'Shopping Bag'

4.2. Use-Case: Check Shopping Cart

1. Actor: User
2. Preconditions: User has an existing instance
3. Parameters: Session ID
4. Postconditions: None
5. Result: The products contained in the shopping cart
6. Actions:
    1. User: Requests shopping cart current product catalog
    2. System: Retrieves product specifications from each 'Shopping Bag'

4.3. Use-Case: Remove From Shopping Cart

1. Actor: User
2. Preconditions:
    1. User has an existing instance
    2. ‘Shopping Cart’ contains at least 1 product
3. Parameters:
    1. Session ID
    2. Product ID
4. Postconditions: User’s shopping cart does not contain the product
5. Result: None
6. Actions:
    1. User: Requests a product be removed from his shopping cart
    2. System: Removes product from the ‘Shopping Bag’ representing the shop ID

4.4. Use-Case: Edit Product Specifications In Shopping Cart

1. Actor: User
2. Preconditions:
    1. User has an existing instance
    2. ‘Shopping Cart’ contains at least 1 product
    3. Desired product quantity is within the shop’s stock
3. Parameters:
    1. Session ID
    2. Product ID
    3. Product Quantity
    4. Additional product modification details
4. Postconditions: User’s shopping cart’s content reflects changes
5. Result: None
6. Actions:
    1. User: Requests product modification of a product in shopping cart
    2. System: Checks that the product quantity does not exceed the shop’s stock
    3. System: Modifies product according to request

5. Use-Case: Checkout

1. Actor: User
2. Preconditions: User has at least one product in shopping cart
3. Parameters:
    1. Session ID
    2. Payment Details
    3. Delivery Details
4. Postconditions:
    Success Scenario:
        1. 'Shopping Cart' is empty
        2. Products in all shops have their quantity adjusted accordingly
        3. Order details are stored in the database
    Failure Scenario:
        1. 'Shopping Cart' is unchanged
        2. Product quantity in all shops is unchanged
5. Result: Notification of successful purchase
6. Actions:
    1. User: Requests transaction finalization
    2. User: Inputs payment and delivery details
    3. System: Initiates product quantity modification
        i. System: In case of failure the user is informed, and the process is aborted
    4. System: Checks that all cart bags are within their respective shops' policies.
    5. System: Checks for and sets the prices of all products according to on-going discounts.
    6. System: Creates an 'Order' with the given parameters
    7. System: Calls 'Payment Service' to confirm transaction validity
        i. System: If system receives a negative response from the service, user is informed, and the process is aborted
        ii. System: Initiates rollback (original product quantities are restored) using the newly created 'Order'
    8. System: Receives a positive response from the payment service
    9. System: Calls 'Delivery Service' to initiate product shipment
        i. System: If system receives a negative response from the service, user is informed, and the process is aborted
        ii. System: Initiates rollback (original product quantities are restored) using the newly created 'Order'
    10. System: Saves the successful order details in the database
    11. System: Notifies listeners interested in successful purchase completion (Initiates real-time and delayed notification processes)
    12. System: Notifies user of successful purchase

5.1 Use-Case: Check Product Availability In Shop (Product Purchase)

1. Actor: System
2. Preconditions: None
3. Parameters:
    1. Shop ID
    2. Product ID
4. Postconditions: None
5. Result: Returns the product's availability in the shop
6. Actions:
    1. System: Accesses specified shop
    2. System: Accesses specified product in shop
    3. System: Returns a response containing the remaining quantity (if any), or the unavailability of the product (e.g. the product was removed from the shop)

5.2 Use-Case: Stock Management Modification (Product Purchase)

1. Actor: System
2. Preconditions: None
3. Parameters:
    1. Shop ID
    2. Product ID
    3. Product quantity
4. Postconditions:
    Success Scenario: The specified product's quantity is modified
    Failure Scenario: None
5. Result: Returns whether the process has been successful
6. Actions:
    1. System: Checks for product availability in the shop
    2. System: If product quantity is insufficient for the desired operation, abort
    3. System: Modifies product quantity available for purchase according to the given amount
    4. System: Returns modification result

5.3 Use-Case: Create Order

1. Actor: System
2. Preconditions: None
3. Parameters: Shopping Cart
4. Postconditions: A new ‘Buyer Order’ and respective ‘Shop Orders’ exist in the system
5. Result: None
6. Actions:
    1. System: Create a new ‘Buyer Order’ instance
    2. System: Create new ‘Shop Order’ instances per shopping bag in the given shopping cart
    3. System: Fix all product related prices (according to shops’ policies and discounts) in the shop orders
    4. System: Add a timestamp representing the time of ‘Checkout’ to the ‘Buyer Order’ instance

Member Use-Cases:

3. General Member Use-Cases:

0. Use-Case: Exit Marketplace (Member)

    1. Actor: Member
    2. Preconditions: User is logged in
    3. Parameters: Session ID
    4. Postconditions:
        1. User is logged out
        2. 'Shopping Cart', as well as other member specific details are preserved
    5. Result: User is no longer able to perform marketplace related actions
    6. Actions:
        1. User: Requests to leave the marketplace
        2. System: Logs user out
        3. System: Closes marketplace system instance

1. Use-Case: Logout

    1. Actor: Member
    2. Preconditions: User is logged in
    3. Parameters:
        1. Username
        2. Session ID
    4. Postconditions:
        1. User is not logged in
        2. 'Guest' instance representing the user exists
        3. 'Guest' instance has an empty shopping cart
        4. 'Guest' instance is associated with the user
    5. Result: User is associated with a 'Guest' instance
    6. Actions:
        1. User: Requests to log out
        2. System: Marks associated 'Member' instance as logged out
        3. System: Creates a new 'Guest' instance with an empty shopping cart
        4. System: Presents to the user relevant guest actions and data

Member Payment Use-Cases:

2. Use-Case: Set Up Shop

1. Actor: Member
2. Preconditions: User is logged in
3. Parameters:
    1. Session ID
    2. Username
    3. Shop Name
4. Postconditions:
    1. A 'Shop' instance exists
    2. The 'Shop' instance is associated with the 'Member' as its founder using his ID
    3. The 'Shop' is active
    4. The 'Member' is assigned the 'Shop Owner' role of the create shop
5. Result: The user can now perform shop related actions as its founder
6. Actions:
    1. User: Requests to open a new shop
    2. System: Creates a new 'Shop' instance with the user as its founder, the given name and sets it as an active shop

4. Shop Owner Use-Cases:

1.1. Use-Case: Stock Management (Product Addition)

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner or manager with sufficient permissions of the given shop
    3. Product does not exist in the shop
    4. Product quantity is positive
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
    4. Product Category
    5. Product Name
    6. Product Price
    7. Product Quantity
    8. Product Description (optional)
4. Postconditions: Specified product is associated with the given shop
5. Result: None
6. Actions:
    1. User: Requests to add a product to the shop
    2. System: Adds the product to the shop with the specified details

1.2. Use-Case: Stock Management (Product Removal)

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner or manager with sufficient permissions of the given shop
    3. Product exists in shop
    4. A 'Checkout' with the desired product is not taking place
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
    4. Product ID
4. Postconditions: Product does not exist in shop
5. Result: None
6. Actions:
    1. User: Requests to remove a product from the shop
    2. System: Removes product listing from the shop

1.3. Use-Case: Stock Management (Product Modification)

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner or manager with sufficient permissions of the given shop
    3. Product exists in shop
    4. Product quantity is positive
    5. A 'Checkout' with the desired product is not taking place
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
    4. Product ID
    5. Product Quantity
4. Postconditions: The specified product's quantity is modified
5. Result: None
6. Actions:
    1. User: Requests to modify a product's quantity in the shop
    2. System: Adds the product to the shop with the specified quantity

2.1.1 Use-Case: Add Shop Policies

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner or manager with sufficient permissions of the given shop
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
    4. Policy Details
4. Postconditions: Policy is added to the shop
5. Result: None
6. Actions:
    1. User: Requests to modify a shop's policies
    2. System: Shop policy is adjusted according to the specified details

2.1.2 Use-Case: Modify Shop's Policies

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner or manager with sufficient permissions of the given shop
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
    4. Policy ID
    5. Policy Type
    6. Predicate Info
4. Postconditions: Shop policy is modified according to the details specified
5. Result: None
6. Actions:
    1. User: Requests to modify a shop's policies
    2. System: Shop policy is adjusted according to the specified details

2.1.3 Use-Case: Remove Shop Policies

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner or manager with sufficient permissions of the given shop
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
    4. Policy ID
4. Postconditions: Shop policy is removed from the shop
5. Result: None
6. Actions:
    1. User: Requests to modify a shop's policies
    2. System: Shop policy is adjusted according to the specified details

2.2.1 Use-Case: Add Shop Discount

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner or manager with sufficient permissions in the given shop
3. Parameters:
    1. Username
    2. Shop ID
    3. Discount Percent
    4. Discount Type (Product, Category or Shop)
    5. Discount Rule (Simple, Conditional, Complex)
    6. Predicates (optional)
4. Postconditions: A new shop discount is added according to the details specified
5. Result: Return discount ID
6. Actions:
    1. User: Requests to add a discount to a shop with shop ID while specifying all relevant details
    2. System: Ensures member is logged in
    3. System: Checks for sufficient permissions
    4. System: Adds the new shop discount to the specified shop

2.2.2 Use-Case: Modify Shop Discount

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner or manager with sufficient permissions in the given shop
    3. A discount with the given ID exists
3. Parameters:
    1. Username
    2. Shop ID
    3. Discount ID
    4. Discount Percent
    5. Discount Type (Product, Category or Shop)
    6. Predicates (optional)
4. Postconditions: The specified shop discount is modified according to the given parameters
5. Result: None
6. Actions:
    1. User: Requests to modify a discount to a shop with shop ID while specifying all relevant details
    2. System: Ensures member is logged in
    3. System: Checks for sufficient permissions
    4. System: Modifies the desired shop discount in the specified shop if it exists

2.2.3 Use-Case: Remove Shop Discount

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner or manager with sufficient permissions in the given shop
    3. A discount with the given ID exists
3. Parameters:
    1. Username
    2. Shop ID
    3. Discount ID
4. Postconditions: The specified shop discount does not exist in the shop
5. Result: None
6. Actions:
    1. User: Requests to remove a discount from a shop with shop ID
    2. System: Ensures member is logged in
    3. System: Checks for sufficient permissions
    4. System: Removes the desired shop discount from the specified shop if it exists

4. Use-Case: Appoint Shop Owner

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner of the given shop
    3. Appointed user is a member and not a shop owner of the given shop
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
    4. Appointed member username
    5. Title (optional)
4. Postconditions:
    1. Appointed member is associated with a 'Shop Owner ' role of the given shop ID
    2. The user is assigned as the appointed member's unique nominator
5. Result: Appointed user can now perform shop owner operations
6. Actions:
    1. User: Requests the nomination of a member to 'Shop Owner'
    2. System: Assigns member the 'Shop Owner' state of the shop
    3. System: Assigns the user as the member's unique nominator

6. Use-Case: Appoint Shop Manager

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner of the given shop
    3. Appointed user is a member and not a shop owner or manager of the given shop
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
    4. Appointed member username
    5. Title (optional)
4. Postconditions:
    1. Appointed member is a shop manager of the given shop ID
    2. The user is assigned as the appointed member's unique nominator
5. Result: Appointed user can now perform shop manager operations
6. Actions:
    1. User: Requests the nomination of a member to 'Shop Manager'
    2. System: Assigns member the 'Shop Manager' state of the shop
    3. System: Assigns the user as the member's unique nominator

7.1. Use-Case: Add Shop Manager Permissions

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is the appointing shop owner of the given shop
    3. Respective user is a shop manager of the shop
3. Parameters:
    1. Session ID
    2. Username
    3. Shop manager's username
    4. Shop ID
    5. Permissions
4. Postconditions: Shop manager has the specified permissions selected
5. Result: Shop manager can perform actions requiring specified permissions
6. Actions:
    1. User: Specifies shop to manage
    2. User: Specifies the shop manager to add permissions to
    3. User: Specifies permissions to add
    4. System: Modifies 'Shop Manager' state permissions

7.2. Use-Case: Remove Shop Manager Permissions

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is the appointing shop owner of the given shop
    3. Respective user is a shop manager of the given shop
3. Parameters:
    1. Session ID
    2. Username
    3. Shop manager's username
    4. Shop ID
    5. Permissions
4. Postconditions: Shop manager cannot perform actions requiring specified permissions
5. Result: Shop manager cannot perform actions requiring specified permissions
6. Actions:
    1. User: Specifies shop to manage
    2. User: Specifies the shop manager to remove permissions from
    3. User: Specifies permissions to remove
    4. System: Modifies 'Shop Manager' state permissions

9. Use-Case: Close Shop

1. Actor: Shop Founder
2. Preconditions:
    1. User is logged in
    2. User is the shop founder of the given shop
    3. Shop is open
    4. A 'Checkout' with products in the shop is not taking place
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
4. Postconditions:
    1. Shop status is inactive (regular members are unable to get information regarding the shop and its products)
    2. Existing shop owners and managers retain their status
5. Result: Shop owners and managers receive a notification regarding the action
6. Actions:
    1. User: Specifies shop to close
    2. System: Waits for any ongoing checkout requests to conclude (successfully or not)
    3. System: Sets shop's status to inactive

11. Use-Case: Request Shop Personnel Info

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner of the given shop
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
4. Postconditions: Shop manager cannot perform actions requiring specified permissions
5. Result: The system displays information regarding the shop's personnel as well as the shop managers' permissions
6. Actions:
    1. User: Specifies shop to inspect
    2. System: Retrieves list of shop managers and owners
    3. System: Retrieves list of personnel permissions

13. Use-Case: Get Shop Purchase History (Shop Owner)

1. Actor: Member
2. Preconditions:
    1. User is logged in
    2. User is a shop owner of the specified shop
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
    4. Time interval
    5. Filter details (optional)
4. Postconditions: None
5. Result: System displays product purchase history (retaining all original details)
6. Actions:
    1. User: Specifies shop to inspect
    2. User: Requests purchase history
    3. User: Specifies time interval between which to search
    4. User: Specifies search filters
    5. System: Retrieves list of transactions

5. Shop Manager Use-Cases:

All operations according to given permissions

6. System Admin Use-Cases:

0. Use-Case: Register (Admin)

1. Actor: System Admin
2. Preconditions: A 'Member' with the same username does not exist in the system
3. Parameters:
    1. Session ID
    2. Username
    3. Password
    4. Identifying details
4. Postconditions:
    1. New 'Member' instance exists associated with the given username
    2. The 'System Admin' role is associated with the new instance
    3. The new 'System Admin' instance holds all identifying details given by the system admin
5. Result: A new 'System Admin' is added to the system
6. Actions:
    1. System Admin: Inputs all relevant identifying details
    2. System Admin: Confirms input
    3. System: Checks for data validity
        i. System: Finds that data is invalid
            a. System: Presents error message
        ii. System: Finds that data is valid
            a. System: Create new 'Member' instance with the given identifying details
            b. System: Associates 'Member' with a 'System Admin' role

4. Use-Case: Get Shop Purchase History (Admin)

1. Actor: System Admin
2. Preconditions:
    1. User is logged in
    2. User is a 'System Admin'
3. Parameters:
    1. Session ID
    2. Username
    3. Shop ID
    4. Time interval
    5. Filter details (optional)
4. Postconditions: None
5. Result: System displays product purchase history (retaining all original details)
6. Actions:
    1. System Admin: Specifies shop to inspect
    2. System Admin: Requests purchase history
    3. System Admin: Specifies time interval between which to search
    4. System Admin: Specifies search filters
    5. System: Retrieves list of transactions