***I. System Use-Cases:***

1. Use-Case: Initialize Marketplace System

1. Actor: System Admin
2. Preconditions: None
3. Parameters: Admin Credentials
4. Postconditions:
5. [‘System Admin’ is initialized](#Register_Admin)

2. System has [established connections with external services](#Add_connection_with_an_external_service) (payment, delivery)

1. Result: Marketplace system is initialized and ready for use
2. Actions:
3. System Admin: Runs the marketplace process
4. System: [Registers a ‘System Admin’ member](#Register_Admin)
5. System: Associates System Admin with ‘System Admin’ instance
6. System: [Establishes connections with payment and delivery services](#Add_connection_with_an_external_service)

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

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

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

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

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

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

3. Use-Case: Call Payment Service

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

4. Use-Case: Call Delivery Service

1. Actor: System
2. Preconditions:
3. A [checkout](#Checkout) operation has been performed by a user
4. A [payment service](#Call_Payment_Service) has confirmed the transaction
5. A connection with a delivery service exists
6. Parameters:
7. Delivery details
8. Client credentials
9. Postconditions: None
10. Result: Delivery request confirmation/refusal
11. Actions:
12. System: Forwards order details to external service
13. System: Receives external service response
14. System: Returns response

5. Use-Case: Real-Time Notifications

1. Actor: System
2. Preconditions: Users are [logged in](#Login)
3. Parameters:
4. Usernames
5. Condition/Message
6. Postconditions: All users related to the satisfied conditions have a pending message
7. Result: None
8. Actions:
9. System: Creates a message according to the satisfied condition
10. System: Notifies all usernames a message is pending

6. Use-Case: Delayed Notifications

1. Actor: System
2. Preconditions: Users are [logged out](#Logout)
3. Parameters:
4. Usernames
5. Condition/Message
6. Postconditions: Database contains messages destined for the specified users
7. Result: None
8. Actions:
9. System: Creates a message according to the satisfied condition
10. 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](#Checkout)
* [A shop is closed](#Close_Shop)
* A shop is re-opened
* A user nomination has been rescinded
* A user received a message/inquiry

1. Parameters:
2. Usernames
3. Condition/Message
4. Postconditions: All users related to the satisfied conditions have a pending message
5. Result: None
6. Actions:
7. System: Creates a message according to the satisfied condition
8. System: Calls [Real-Time Notifications](#Real_Time_Notifications) for logged in members, calls [Delayed Notifications](#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:
5. ‘Guest’ instance representing the user exists
6. ‘Guest’ instance has an empty shopping cart
7. ‘Guest' instance is associated with the user
8. Result: User can perform [general and purchase related actions](#User_Related_Use_Cases)
9. Actions:
10. System: Creates a new ‘Guest’ instance with an empty shopping cart
11. 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:
5. ‘Shopping Cart’ is emptied
6. ‘Guest’ instance is deleted
7. Result: User can no longer perform any actions within the system
8. Actions:
9. System: Empties the ‘Shopping Cart’
10. System: Deletes the associated ‘Guest’ instance
11. System: Closes marketplace system instance

3. Use-Case: Register

1. Actor: User
2. Preconditions:
3. ‘Guest’ instance associated with the user exists
4. A ‘Member’ with the same username does not exist in the system
5. Parameters:
6. Session ID
7. Username
8. Password
9. Postconditions:
10. New ‘Member’ instance exists
11. The new ‘Member’ instance holds all identifying details given by the user
12. Result: A new ‘Member’ is added to the system
13. Actions:
14. User: Inputs all relevant identifying details
15. User: Confirms input
16. System: Checks for data validity
17. System: Finds that data is invalid
    1. System: Present error message
18. System: Finds that data is valid
19. 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:
4. Session ID
5. Username
6. Password
7. Postconditions:
8. User is identified as ‘Member’ with its associated details
9. User is associated with his unique shopping cart
10. Result: User can perform any [member related operations](#Member_Use_Cases)
11. Actions:
12. User: Inputs username
13. User: Inputs password
14. User: Confirms input
15. System: Checks for data validity
16. System: Finds that data is invalid
    1. System: Presents error message
17. System: Finds that data is valid
18. 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:
4. Session ID
5. Shop ID
6. Postconditions: None
7. Result: Display relevant shop info, including products that the shop is offering
8. Actions:
9. User: Requests shop details
10. System: Verifies user has accessed the marketplace (guest) or is logged in (member)
11. System: Searches for shop
12. System: Finds that shop exists
13. System: Displays relevant shop info
14. System: Finds that shop doesn’t exist
15. 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:
4. Session ID
5. Keywords and filters
6. Postconditions: None
7. Result: Products corresponding to the given parameters
8. Actions:
9. System: Initialize search process
10. User: Inputs keywords
11. User: Inputs filters (Optional)
12. User: Confirms input
13. System: Searches according to the given parameters
14. 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:
3. User has an existing instance
4. The user is the owner of the shopping cart
5. A shop with the shop ID exists
6. A product with the product ID exists in the relevant shop
7. Desired product quantity is within the shop’s stock
8. Parameters:
9. Session ID
10. Product ID
11. Product Quantity
12. Postconditions: User’s shopping cart contains the corresponding product
13. Result: None
14. Actions:
15. User: Selects product to add
16. System: Checks if product with product ID exists
17. System: Checks that the product quantity does not exceed the shop’s stock
18. 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:
7. User: Requests shopping cart current product catalog
8. System: Retrieves product specifications from each ‘Shopping Bag’

4.3. Use-Case: Remove From Shopping Cart

1. Actor: User
2. Preconditions:
3. User has an existing instance
4. ‘Shopping Cart’ contains at least 1 product
5. Parameters:
6. Session ID
7. Product ID
8. Postconditions: User’s shopping cart does not contain the product
9. Result: None
10. Actions:
11. User: Requests a product be removed from his shopping cart
12. 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:
3. User has an existing instance
4. ‘Shopping Cart’ contains at least 1 product
5. Desired product quantity is within the shop’s stock
6. Parameters:
7. Session ID
8. Product ID
9. Product Quantity
10. Additional product modification details
11. Postconditions: User’s shopping cart’s content reflects changes
12. Result: None
13. Actions:
14. User: Requests product modification of a product in shopping cart
15. System: Checks that the product quantity does not exceed the shop’s stock
16. 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:
4. Session ID
5. Payment Details
6. Delivery Details
7. 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
3. Result: Notification of successful purchase
4. Actions:
5. User: Requests transaction finalization
6. User: Inputs payment and delivery details
7. System: [Initiates product quantity modification](#Stock_Management_Modification_Purchase)
8. System: In case of failure the user is informed, and the process is aborted
9. System: Checks that all cart bags are within their respective shops’ policies.
10. System: Checks for and sets the prices of all products according to on-going discounts.
11. System: [Creates an ‘Order’](#Create_Order) with the given parameters
12. System: [Calls ‘Payment Service’](#Call_Payment_Service) to confirm transaction validity
13. System: If system receives a negative response from the service, user is informed, and the process is aborted
14. System: [Initiates rollback (original product quantities are restored)](#Stock_Management_Modification_Purchase) using the newly created ‘Order’
15. System: Receives a positive response from the payment service
16. System: [Calls ‘Delivery Service’](#Call_Delivery_Service) to initiate product shipment
17. System: If system receives a negative response from the service, user is informed, and the process is aborted
18. System: [Initiates rollback (original product quantities are restored)](#Stock_Management_Modification_Purchase) using the newly created ‘Order’
19. System: Saves the successful order details in the database
20. System: Notifies listeners interested in successful purchase completion ([Initiates real-time](#Real_Time_Notifications) and [delayed notification](#Delayed_Notifications) processes)
21. 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:
4. Shop ID
5. Product ID
6. Postconditions: None
7. Result: Returns the product’s availability in the shop
8. Actions:
9. System: Accesses specified shop
10. System: Accesses specified product in shop
11. 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:
4. Shop ID
5. Product ID
6. Product quantity
7. Postconditions:

Success Scenario: The specified product’s quantity is modified

Failure Scenario: None

1. Result: Returns whether the process has been successful
2. Actions:
3. System: [Checks for product availability in the shop](#Check_Product_Availability_In_Shop)
4. System: If product quantity is insufficient for the desired operation, abort
5. System: Modifies product quantity available for purchase according to the given amount
6. 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:
7. System: Create a new ‘Buyer Order’ instance
8. System: Create new ‘Shop Order’ instances per shopping bag in the given shopping cart
9. System: Fix all product related prices (according to shops’ policies and discounts) in the shop orders
10. 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](#Login)
3. Parameters: Session ID
4. Postconditions:
5. User is [logged out](#Logout)
6. ‘Shopping Cart’, as well as other member specific details are preserved
7. Result: User is no longer able to perform marketplace related actions
8. Actions:
9. User: Requests to leave the marketplace
10. System: [Logs user out](#Logout)
11. System: Closes marketplace system instance

1. Use-Case: Logout

1. Actor: Member
2. Preconditions: User is [logged in](#Login)
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](#Login)
3. Parameters:
4. Session ID
5. Username
6. Shop Name
7. Postconditions:
8. A ‘Shop’ instance exists
9. The ‘Shop’ instance is associated with the ‘Member’ as its founder using his ID
10. The ‘Shop’ is active
11. The ‘Member’ is assigned the ‘Shop Owner’ role of the create shop
12. Result: The user can now perform shop related actions as its founder
13. Actions:
14. User: Requests to open a new shop
15. 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:
3. User is [logged in](#Login)
4. User is a shop owner or manager with sufficient permissions of the given shop
5. Product does not exist in the shop
6. Product quantity is positive
7. Parameters:
8. Session ID
9. Username
10. Shop ID
11. Product Category
12. Product Name
13. Product Price
14. Product Quantity
15. Product Description (optional)
16. Postconditions: Specified product is associated with the given shop
17. Result: None
18. Actions:
19. User: Requests to add a product to the shop
20. System: Adds the product to the shop with the specified details

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

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner or manager with sufficient permissions of the given shop
5. Product exists in shop
6. A ‘[Checkout](#Checkout)’ with the desired product is not taking place
7. Parameters:
8. Session ID
9. Username
10. Shop ID
11. Product ID
12. Postconditions: Product does not exist in shop
13. Result: None
14. Actions:
15. User: Requests to remove a product from the shop
16. System: Removes product listing from the shop

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

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner or manager with sufficient permissions of the given shop
5. Product exists in shop
6. Product quantity is positive
7. A ‘[Checkout](#Checkout)’ with the desired product is not taking place
8. Parameters:
9. Session ID
10. Username
11. Shop ID
12. Product ID
13. Product Quantity
14. Postconditions: The specified product’s quantity is modified
15. Result: None
16. Actions:
17. User: Requests to modify a product’s quantity in the shop
18. System: Adds the product to the shop with the specified quantity

2.1.1 Use-Case: Add Shop Policies

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner or manager with sufficient permissions of the given shop
5. Parameters:
6. Session ID
7. Username
8. Shop ID
9. Policy Details
10. Postconditions: Policy is added to the shop
11. Result: None
12. Actions:
13. User: Requests to modify a shop’s policies
14. System: Shop policy is adjusted according to the specified details

2.1.2 Use-Case: Modify Shop’s Policies

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner or manager with sufficient permissions of the given shop
5. Parameters:
6. Session ID
7. Username
8. Shop ID
9. Policy ID
10. Policy Type
11. Predicate Info
12. Postconditions: Shop policy is modified according to the details specified
13. Result: None
14. Actions:
15. User: Requests to modify a shop’s policies
16. System: Shop policy is adjusted according to the specified details

2.1.3 Use-Case: Remove Shop Policies

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner or manager with sufficient permissions of the given shop
5. Parameters:
6. Session ID
7. Username
8. Shop ID
9. Policy ID
10. Postconditions: Shop policy is removed from the shop
11. Result: None
12. Actions:
13. User: Requests to modify a shop’s policies
14. System: Shop policy is adjusted according to the specified details

2.2.1 Use-Case: Add Shop Discount

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

2.2.2 Use-Case: Modify Shop Discount

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner or manager with sufficient permissions in the given shop
5. A discount with the given ID exists
6. Parameters:
7. Username
8. Shop ID
9. Discount ID
10. Discount Percent
11. Discount Type (Product, Category or Shop)
12. Predicates (optional)
13. Postconditions: The specified shop discount is modified according to the given parameters
14. Result: None
15. Actions:
16. User: Requests to modify a discount to a shop with shop ID while specifying all relevant details
17. System: Ensures member is logged in
18. System: Checks for sufficient permissions
19. 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:
3. User is [logged in](#Login)
4. User is a shop owner or manager with sufficient permissions in the given shop
5. A discount with the given ID exists
6. Parameters:
7. Username
8. Shop ID
9. Discount ID
10. Postconditions: The specified shop discount does not exist in the shop
11. Result: None
12. Actions:
13. User: Requests to remove a discount from a shop with shop ID
14. System: Ensures member is logged in
15. System: Checks for sufficient permissions
16. System: Removes the desired shop discount from the specified shop if it exists

4. Use-Case: Appoint Shop Owner

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner of the given shop
5. Appointed user is a member and not a shop owner of the given shop
6. Parameters:
7. Session ID
8. Username
9. Shop ID
10. Appointed member username
11. Title (optional)
12. Postconditions:
13. Appointed member is associated with a ‘Shop Owner ‘ role of the given shop ID
14. The user is assigned as the appointed member’s unique nominator
15. Result: Appointed user can now perform shop owner operations
16. Actions:
17. User: Requests the nomination of a member to ‘Shop Owner’
18. System: Assigns member the ‘Shop Owner’ state of the shop
19. System: Assigns the user as the member’s unique nominator

6. Use-Case: Appoint Shop Manager

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner of the given shop
5. Appointed user is a member and not a shop owner or manager of the given shop
6. Parameters:
7. Session ID
8. Username
9. Shop ID
10. Appointed member username
11. Title (optional)
12. Postconditions:
13. Appointed member is a shop manager of the given shop ID
14. The user is assigned as the appointed member’s unique nominator
15. Result: Appointed user can now perform shop manager operations
16. Actions:
17. User: Requests the nomination of a member to ‘Shop Manager’
18. System: Assigns member the ‘Shop Manager’ state of the shop
19. System: Assigns the user as the member’s unique nominator

7.1. Use-Case: Add Shop Manager Permissions

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is the appointing shop owner of the given shop
5. Respective user is a shop manager of the shop
6. Parameters:
7. Session ID
8. Username
9. Shop manager’s username
10. Shop ID
11. Permissions
12. Postconditions: Shop manager has the specified permissions selected
13. Result: Shop manager can perform actions requiring specified permissions
14. Actions:
15. User: Specifies shop to manage
16. User: Specifies the shop manager to add permissions to
17. User: Specifies permissions to add
18. System: Modifies ‘Shop Manager’ state permissions

7.2. Use-Case: Remove Shop Manager Permissions

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is the appointing shop owner of the given shop
5. Respective user is a shop manager of the given shop
6. Parameters:
7. Session ID
8. Username
9. Shop manager’s username
10. Shop ID
11. Permissions
12. Postconditions: Shop manager cannot perform actions requiring specified permissions
13. Result: Shop manager cannot perform actions requiring specified permissions
14. Actions:
15. User: Specifies shop to manage
16. User: Specifies the shop manager to remove permissions from
17. User: Specifies permissions to remove
18. System: Modifies ‘Shop Manager’ state permissions

9. Use-Case: Close Shop

1. Actor: Shop Founder
2. Preconditions:
3. User is [logged in](#Login)
4. User is the shop founder of the given shop
5. Shop is open
6. A ‘[Checkout](#Checkout)’ with products in the shop is not taking place
7. Parameters:
8. Session ID
9. Username
10. Shop ID
11. Postconditions:
12. Shop status is inactive (regular members are unable to get information regarding the shop and its products)
13. Existing shop owners and managers retain their status
14. Result: Shop owners and managers receive a notification regarding the action
15. Actions:
16. User: Specifies shop to close
17. System: Waits for any ongoing checkout requests to conclude (successfully or not)
18. System: Sets shop’s status to inactive

11. Use-Case: Request Shop Personnel Info

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner of the given shop
5. Parameters:
6. Session ID
7. Username
8. Shop ID
9. Postconditions: Shop manager cannot perform actions requiring specified permissions
10. Result: The system displays information regarding the shop’s personnel as well as the shop managers’ permissions
11. Actions:
12. User: Specifies shop to inspect
13. System: Retrieves list of shop managers and owners
14. System: Retrieves list of personnel permissions

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

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner of the specified shop
5. Parameters:
6. Session ID
7. Username
8. Shop ID
9. Time interval
10. Filter details (optional)
11. Postconditions: None
12. Result: System displays product purchase history (retaining all original details)
13. Actions:
14. User: Specifies shop to inspect
15. User: Requests purchase history
16. User: Specifies time interval between which to search
17. User: Specifies search filters
18. 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:
4. Session ID
5. Username
6. Password
7. Identifying details
8. Postconditions:
9. New ‘Member’ instance exists associated with the given username
10. The ‘System Admin’ role is associated with the new instance
11. The new ‘System Admin’ instance holds all identifying details given by the system admin
12. Result: A new ‘System Admin’ is added to the system
13. Actions:
14. System Admin: Inputs all relevant identifying details
15. System Admin: Confirms input
16. System: Checks for data validity
17. System: Finds that data is invalid
    1. System: Presents error message
18. System: Finds that data is valid
19. System: Create new ‘Member’ instance with the given identifying details
20. System: Associates ‘Member’ with a ‘System Admin’ role

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

1. Actor: System Admin
2. Preconditions:
3. User is [logged in](#Login)
4. User is a ‘System Admin’
5. Parameters:
6. Session ID
7. Username
8. Shop ID
9. Time interval
10. Filter details (optional)
11. Postconditions: None
12. Result: System displays product purchase history (retaining all original details)
13. Actions:
14. System Admin: Specifies shop to inspect
15. System Admin: Requests purchase history
16. System Admin: Specifies time interval between which to search
17. System Admin: Specifies search filters
18. System: Retrieves list of transactions