***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. Parameters: Modification details
6. Postconditions:
7. Current marketplace state (i.e. state has not been altered)
8. The connection with the external service has updated parameters
9. Result: All traffic related to the external service is routed to it according to the parameters specified
10. Actions:
11. System Admin: Specifies modification details
12. System: Forwards request to external service
13. 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. 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 original external service does not exist
11. An active connection with the new external service exists
12. Result: All traffic related to the external service is routed to the new service
13. Actions:
14. System Admin: Specifies external service to connect to
15. System Admin: Specifies relevant details to allow connection
16. System: Disconnects from original external service
17. 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: None
3. Parameters: None
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: Guest ID/Username
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: Identifying details
6. Postconditions:
7. New ‘Member’ instance exists
8. The new ‘Member’ instance holds all identifying details given by the user
9. Result: A new ‘Member’ is added to the system
10. Actions:
11. User: Inputs all relevant identifying details
12. User: Confirms input
13. System: Checks for data validity
14. System: Finds that data is invalid
    1. System: Present error message
15. System: Finds that data is valid
16. 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. Username
5. Password
6. Postconditions:
7. User is identified as ‘Member’ with its associated details
8. User is associated with his unique shopping cart
9. [Listeners were notified](#Delayed_Notifications) that the user has logged in and can receive notifications
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
19. System: Notifies listeners of the user’s successful login attempt

**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. Guest ID/Username
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. Guest ID/Username
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. Username\Guest ID
10. Shop ID
11. Product ID
12. Product Quantity
13. Postconditions: User’s shopping cart contains the corresponding product
14. Result: None
15. Actions:
16. User: Selects shop to browse
17. System: Checks if a shop with shop ID exists
18. User: Selects product from shop to add
19. System: Checks if product with product ID exists
20. System: Checks that the product quantity does not exceed the shop’s stock
21. 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: Username\Guest 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. Username\Guest ID
7. Product ID
8. Shop ID
9. Postconditions: User’s shopping cart does not contain the product
10. Result: None
11. Actions:
12. User: Requests a product be removed from his shopping cart
13. 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. Username\Guest ID
8. Shop ID
9. Product ID
10. Product Quantity
11. Additional product modification details
12. Postconditions: User’s shopping cart’s content reflects changes
13. Result: None
14. Actions:
15. User: Requests product modification from a product in shopping cart
16. System: Checks that the product quantity does not exceed the shop’s stock
17. 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. User’s ‘Shopping Cart’
5. Payment Details
6. Delivery Details (optional)
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: [Creates an ‘Order’](#Create_Order) with the given parameters
10. System: [Calls ‘Payment Service’](#Call_Payment_Service) to confirm transaction validity
11. System: If system receives a negative response from the service, user is informed, and the process is aborted
12. System: [Initiates rollback (original product quantities are restored)](#Stock_Management_Modification_Purchase) using the newly created ‘Order’
13. System: Receives a positive response from the payment service
14. System: [Calls ‘Delivery Service’](#Call_Delivery_Service) to initiate product shipment
15. System: If system receives a negative response from the service, user is informed, and the process is aborted
16. System: [Initiates rollback (original product quantities are restored)](#Stock_Management_Modification_Purchase) using the newly created ‘Order’
17. System: Saves the successful order details in the database
18. System: Notifies listeners interested in successful purchase completion ([Initiates real-time](#Real_Time_Notifications) and [delayed notification](#Delayed_Notifications) processes)
19. 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: Username
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: Username
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. Username
5. Shop Name
6. Purchase and Discount types
7. Purchase and Discount policy details
8. Postconditions:
9. A ‘Shop’ instance exists
10. The ‘Shop’ instance is associated with the ‘Member’ as its founder using his ID
11. The ‘Shop’ is active
12. Purchase and discount types are defined
13. Purchase and discount policies are defined
14. The ‘Member’ is assigned the ‘Shop Owner’ role of the create shop
15. Result: The user can now perform shop related actions as its founder
16. Actions:
17. User: Requests to open a new shop
18. System: Creates a new ‘Shop’ instance with the user as its founder, the given name and sets it as an active shop
19. System: Defines purchase and discount types
20. System: Defines purchase and discount policies

**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
5. Product does not exist in the shop
6. Product quantity is positive
7. Parameters:
8. Username
9. Shop ID
10. Product ID
11. Product quantity
12. Postconditions: Specified product is associated with the given shop
13. Result: None
14. Actions:
15. User: Requests to add a product to the shop
16. System: Adds the product to the shop with the specified quantity

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
5. Product exists in shop
6. A ‘[Checkout](#Checkout)’ with the desired product is not taking place
7. Parameters:
8. Username
9. Shop ID
10. Product ID
11. Postconditions: Product does not exist in shop
12. Result: None
13. Actions:
14. User: Requests to remove a product from the shop
15. System: Waits for any ongoing checkout requests to conclude (successfully or not)
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
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. Username
10. Shop ID
11. Product ID
12. Product quantity
13. Postconditions: The specified product’s quantity is modified
14. Result: None
15. Actions:
16. User: Requests to modify a product’s quantity in the shop
17. System: Waits for any ongoing checkout requests to conclude (successfully or not)
18. System: Adds the product to the shop with the specified quantity

2. Use-Case: Modify Shop’s Purchase/Sale Types and Policies

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner or manager with sufficient permissions
5. A ‘[Checkout](#Checkout)’ with the desired product is not taking place
6. Parameters:
7. Username
8. Shop ID
9. Policy details
10. Postconditions: Shop policy is modified according to the details specified
11. Result: None
12. Actions:
13. User: Requests to modify a shop’s policies
14. System: Waits for any ongoing checkout requests to conclude (successfully or not)
15. System: Shop policy is adjusted according to the specified details

4. Use-Case: Appoint Shop Owner

1. Actor: Member
2. Preconditions:
3. User is [logged in](#Login)
4. User is a shop owner
5. Appointed user is a member and not a shop owner
6. Parameters:
7. Username
8. Appointed member username
9. Shop ID
10. Postconditions:
11. Appointed member is associated with a ‘Shop Owner ‘ role of the given shop ID
12. The user is assigned as the appointed member’s unique nominator
13. Result: Appointed user can now perform shop owner operations
14. Actions:
15. User: Requests the nomination of a member to ‘Shop Owner’
16. System: Assigns member the ‘Shop Owner’ state of the shop
17. 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
5. Appointed user is a member and not a shop owner or manager
6. Parameters:
7. Username
8. Appointed member username
9. Shop ID
10. Postconditions:
11. Appointed member is a shop manager of the given shop ID
12. The user is assigned as the appointed member’s unique nominator
13. Result: Appointed user can now perform shop manager operations
14. Actions:
15. User: Requests the nomination of a member to ‘Shop Manager’
16. System: Assigns member the ‘Shop Manager’ state of the shop
17. 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 a shop owner
5. Respective user is a shop manager of the shop
6. Parameters:
7. Username
8. Shop manager’s username
9. Shop ID
10. Permissions
11. Postconditions: Shop manager has the specified permissions selected
12. Result: Shop manager can perform actions requiring specified permissions
13. Actions:
14. User: Specifies shop to manage
15. User: Specifies the shop manager to add permissions to
16. User: Specifies permissions to add
17. 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 a shop owner
5. Respective user is a shop manager
6. Parameters:
7. Username
8. Shop manager’s username
9. Shop ID
10. Permissions
11. Postconditions: Shop manager cannot perform actions requiring specified permissions
12. Result: Shop manager cannot perform actions requiring specified permissions
13. Actions:
14. User: Specifies shop to manage
15. User: Specifies the shop manager to remove permissions from
16. User: Specifies permissions to remove
17. 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
5. Shop is open
6. A ‘[Checkout](#Checkout)’ with products in the shop is not taking place
7. Parameters:
8. Username
9. Shop ID
10. Postconditions:
11. Shop status is inactive (regular members are unable to get information regarding the shop and its products)
12. Existing shop owners and managers retain their status
13. Result: Shop owners and managers receive a notification regarding the action
14. Actions:
15. User: Specifies shop to close
16. System: Waits for any ongoing checkout requests to conclude (successfully or not)
17. 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
5. Parameters:
6. Username
7. Shop ID
8. Postconditions: Shop manager cannot perform actions requiring specified permissions
9. Result: The system displays information regarding the shop’s personnel as well as the shop managers’ permissions
10. Actions:
11. User: Specifies shop to inspect
12. System: Retrieves list of shop managers and owners
13. 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. Username
7. Shop ID
8. Time interval
9. Filter details (optional)
10. Postconditions: None
11. Result: System displays product purchase history (retaining all original details)
12. Actions:
13. User: Specifies shop to inspect
14. User: Requests purchase history
15. User: Specifies time interval between which to search
16. User: Specifies search filters
17. 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. Username
5. Password
6. Identifying details
7. Postconditions:
8. New ‘Member’ instance exists associated with the given username
9. The ‘System Admin’ role is associated with the new instance
10. The new ‘System Admin’ instance holds all identifying details given by the system admin
11. Result: A new ‘System Admin’ is added to the system
12. Actions:
13. System Admin: Inputs all relevant identifying details
14. System Admin: 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: Create new ‘Member’ instance with the given identifying details
19. 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. Username
7. Shop ID
8. Time interval
9. Filter details (optional)
10. Postconditions: None
11. Result: System displays product purchase history (retaining all original details)
12. Actions:
13. System Admin: Specifies shop to inspect
14. System Admin: Requests purchase history
15. System Admin: Specifies time interval between which to search
16. System Admin: Specifies search filters
17. System: Retrieves list of transactions