Table of contents

System I

- **Use-Case: Adding an external service 1.1.1-a — Success Scenario**
    1. Actor: system manager
    2. Preconditions:
    3. Parameters: New external service
    4. Postconditions: The system contains the new service
    5. Actions:
        a. **system manager:** Requests to add a new external service to the system.
        b. **system**: Request information regarding new service.
        c. **system manager**: Enter relevant information.
        d. **system**: Check that the new service does not already exist.
        e. **system**: Add new service to list of services.

- **Use-Case: Adding an external service 1.1.1-b — Failure Scenario (Service exists in the system)**
    1. Actor: system manager
    2. Preconditions:
    3. Parameters: Existing external service
    4. Postconditions: The system contains the existing service
    5. Actions:
        a. **system manager:** Requests to add a new external service to the system.
        b. **system**: Request information regarding new service.
        c. **system manager**: Enter relevant information.

d. **system**: Check that the new service does not already exist.
e. **system**: Finds out the service already exists.
f. **system**: Raises an error message to the user.

- **Use-Case: Altering an external service 1.1.2-a — Success Scenario**
    1. Actor: system manager
    2. Preconditions:
    3. Parameters:
        a. Identifier of the service to be altered.
        b. New details for the service.
    4. Postconditions: The service was updated.
    5. Actions:
        a. **system manager:** Requests to alter an existing service in the system.
        b. **system**: prompt system manager for the identifier of the service to be altered and the new details for the service.
        c. **system manager**: enter the identifier of the service to be altered and the new details for the service.
        d. **system**: Look up the service by the service identifier.
        e. **system**: replace current information of the service with new information.

- **Use-Case: Altering an external service 1.1.2-b — Failure Scenario (Service doesn't exist in the system)**
    1. Actor: system manager
    2. Preconditions:
    3. Parameters:
        a. Identifier of the service to be altered.
        b. New details for the service.

4. Postconditions:
5. Actions:
    a. **system manager:** Requests to alter an existing service in the system.
    b. **system**: Prompt system manager for the identifier of the service to be altered and the new details.
    c. **system manager**: Enter the identifier of the service to be altered.
    d. **system**: Look up the service.
    e. **system**: Doesn't find the service, raises an error.

- **Use-Case: Removing an external service 1.1.3-a — Success Scenario**
    1. Actor: system manager.
    2. Preconditions:
    3. Parameters: Identifier of the service to be removed.
    4. Postconditions: service is not available in the system.
    5. Actions:
        a. **system manager:** Requests to remove a service from the system.
        b. **system**: Request identifier of the service to be removed.
        c. **system manager**: Enter identifier of the service to be removed.
        d. **system**: Look up service by given identifier.
        e. **system**: Remove service from list of services.

- **Use-Case: Removing an external service 1.1.3-b — Failure Scenario (Service doesn't exist in the system)**
    1. Actor: system manager.
    2. Preconditions:
    3. Parameters: Identifier of the service to be removed.
    4. Postconditions: service is not available in the system.
    5. Actions:

a. **system manager:** Requests to remove a service from the system.
b. **system**: Request identifier ot the service to be removed.
c. **system manager**: Enter identifier of the service to be removed.
d. **system**: Look up service by the given identifier.
e. **system**: The service does not exist in the system, return an error.

- **Use-Case: Show notification for a logged-in user 1.4-a — Success Scenario**
    1. Actor: member
    2. Preconditions:
    3. Parameters: notification details.
    4. Postconditions:
    5. Expected result: Notification for the user.
    6. Actions:
        a. **system**: Receive a notification for the member.
        b. **system**: Verify that the member is logged-in.
        c. **system**: Send notification to the member.

- **Use-Case: Show notification for a logged-in user 1.4-b — Failed Scenario (User is not logged in)**
    1. Actor: member
    2. Preconditions:
    3. Parameters: notification details.
    4. Postconditions: member's notification is saved in the system.
    5. Expected result:
    6. Actions:
        a. **system**: Receive a notification for the member.
        b. **system**: Verify that the member is logged-in.
        c. **system**: Recognizes member isn't logged in
        d. **system:** saves the notification.

- **Use-Case: Show notifications for returning user 1.5 — Success Scenario**
    1. Actor: member
    2. Preconditions: member received a notification while being logged off.
    3. Parameters:
    4. Postconditions: There are no new notifications to show the user.
    5. Expected result: Notifications that were received while the user was away.
    6. Actions:
        a. **member**: Logs into the system.
        b. **system**: Shows the user the notifications they received while away from the system.

Users II

- **Use-Case: Entering the trading system 2.1.1-a — Success Scenario**
    1. Actor: guest
    2. Preconditions:
    3. Parameters:
    4. Postconditions:
        a. The guest is connected to the system
        b. The guest has an empty shopping cart.
    5. Expected result:
    6. Actions:
        a. **guest**: Enters the system.
        b. **system**: Assigns an empty shopping basket to the new guest.

- **Use-Case: Guest registration to the trading system 2.1.3-a — Success Scenario**
    1. Actor: guest
    2. Preconditions: guest is connected to the system.
    3. Parameters: user credentials.
    4. Postconditions: new user is registered in the system with the given user credentials.
    5. Actions:
        a. **guest**: Request to register to the system.
        b. **system**: Request user credentials from the guest.
        c. **guest**: Enter user credentials.
        d. **system**: Verify that user credentials are valid.
        e. **system**: Creates a new user with the given user credentials.

- **Use-Case: Guest registration to the trading system 2.1.3-b Failed Scenario 🎉(Entered user credentials are not valid)**
    1. Actor: guest
    2. Preconditions: guest is connected to the system.
    3. Parameters: user credentials.
    4. Postconditions:
    5. Actions:
        a. **guest**: Request to register.
        b. **system**: Request user credentials from the guest.
        c. **guest**: Enters user credentials.
        d. **system**: Verify that user credentials are valid.
        e. **system**: Finds out that the user credentials are not valid
        f. **system:** raises an error to the user.

- **Use-Case: Guest login to the trading system 2.1.4-a — Success Scenario**
    1. Actor: guest
    2. Preconditions: guest is connected to the system.
    3. Parameters: user credentials
    4. Postconditions: The guest is logged as the member with the given user credentials.
    5. Actions:
        a. **guest**: Request to login to the system.
        b. **system**: Request user credentials from the guest.
        c. **guest**: Enters user credentials.
        d. **system**: Validates user credentials.
        e. **system**: Logs in the guest as the member with the given user credentials.

- **Use-Case: Guest login to the trading system 2.1.4-b Failure Scenario (User credentials are not valid)**
    1. Actor: guest
    2. Preconditions: guest is connected to the system.
    3. Parameters: user credentials
    4. Postconditions:
    5. Actions:
        a. **guest**: Requests to login to the system.
        b. **system**: Request user credentials from the guest.
        c. **guest**: Enters user credentials.
        d. **system**: Validates user credentials.
        e. **system**: Finds out that entered user credentials are not valid
        f. **system:** raises an error message to the guest.

- **Use-Case: User receives information about stores 2.2.1.1-a — Success Scenario**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters: —
    4. Postconditions: —
    5. Expected result: Relevant information about the stores in the system.
    6. Actions:
        a. **user**: Requests from the system for information about the stores in the system.
        b. **system**: Show the user relevant information about the stores.

- **Use-Case: user receives information regarding products in store 2.2.1.2-a — Success Scenario**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters: store identifier
    4. Postconditions:
    5. Expected result: Relevant information about products of the store with the given store identifier.
    6. Actions:
        a. **user**: Request information regarding products in store.
        b. **system**: Prompt user to enter store identifier.
        c. **user**: Enter store identifier.
        d. **system**: Validate the store identifier.
        e. **system**: Send relevant information regarding products in the store with the given store identifier.

- **Use-Case: user receives information regarding products in store 2.2.1.2-b — Failure Scenario (Store identifier is not valid)**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters: store identifier
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Request information regarding products in store.
        b. **system**: Prompt user to enter store identifier.
        c. **user**: Enter store identifier.
        d. **system**: Validate the store identifier
        e. **system**: Finds out that the given store identifier is not valid
        f. **system:** raises an error.

- **Use-Case: Searching for products 2.2.2.1-a — Success Scenario**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters: search parameters and filters
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Requests to search for products with the given search parameters and filters.
        b. **system**: Looks up all products that pass the filters with the given search parameters.
        c. **system**: Renders the list of passing products to the user.

- **Use-Case: Searching for products 2.2.2.1-b — Failed Scenario(No products that pass the filters)**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters: search parameters and filters
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Requests to search for products with the given search parameters and filters.
        b. **system**: Looks up all products that pass the filters with the given search parameters.
        c. **system**: Finds out that no products pass the filters with the given search parameters.
        d. **system:** Raises an appropriate message.

- **Use-Case: Searching for products in a specific store 2.2.2.2-a — Success Scenario**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
        a. search parameters and filters
        b. store identifier.
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Requests to search for products with the given search parameters and filters in the store matching the store identifier.
        b. **system**: Validates store identifier.

c. **system**: Looks up all products in the store that pass the filters with the given search parameters.
d. **system**: Renders the list of passing products to the user.

- **Use-Case: Searching for products in a specific store 2.2.2.2-b — Failed Scenario (store identifier is invalid)**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
        a. search parameters and filters.
        b. store identifier.
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Requests to search for products with the given search parameters and filters in the store matching the store identifier.
        b. **system**: Validates store identifier.
        c. **system**: Finds out given store identifier is invalid
        d. **system:** raises an error.

- **Use-Case: Searching for products in a specific store 2.2.2.2-c — Failed Scenario (No products pass the filters)**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
        a. search parameters and filters.
        b. store identifier.
    4. Postconditions:
    5. Expected result:

6. Actions:
    a. **user**: Requests to search for products with the given search parameters and filters in the store matching the store identifier.
    b. **system**: Validates store identifier.
    c. **system**: Looks up all products that pass the filters with the given search parameters.
    d. **system**: Finds out that no products pass the filters with the given search parameters in the store with the given store identifier, raises an appropriate message.

- **Use-Case: Adding a product to the store basket 2.2.3-a — Success Scenario**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
        a. product identifier
        b. amount of product.
        c. store identifier.
    4. Postconditions: The specified amount of products with the given product identifier were added to the user's basket for the store with the given store identifier.
    5. Expected result:
    6. Actions:
        a. **user**: Requests to add a product with product identifier to shopping basket of store with store identifier.
        b. **system**: Validates store identifier.
        c. **system**: Validates product identifier.
        d. **system**: Validates amount of product.
        e. **system**: Adds the specified amount of products to the store basket

f. **system:** Updates the quantity of instances in the basket.

- **Use-Case: Adding a product to the store basket 2.2.3-b — Alternative Scenario(Basket of store with given store identifier doesn't exist)**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
        a. product identifier.
        b. amount of product.
        c. store identifier.
    4. Postconditions: A new shopping basket is created for the store with the given store identifier, and the specified amount of products with the given product identifier were added to that basket.
    5. Expected result:
    6. Actions:
        a. **user**: Requests to add a product with product identifier to shopping basket of store with store identifier.
        b. **system**: Validates store identifier
        c. **system**: Validates product identifier.
        d. **system**: Validates amount of product.
        e. **system**: Finds out the user doesn't yet have a basket for the store with the given store identifier.
        f. **system:** creates a new basket for that store.
        g. **system**: Adds the specified amount of products to the store basket.
        h. **system:** updates the quantity of instances in the basket.

- **Use-Case: Adding a product to the store basket 2.2.3-c — Failed Scenario(Amount of product is invalid)**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
        a. product identifier.
        b. amount of product.
        c. store identifier.
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Requests to add a product with product identifier to shopping basket of store with store identifier.
        b. **system**: Validates store identifier
        c. **system**: Validates product identifier.
        d. **system**: Validates amount of product.
        e. **system**: Finds out that the amount of product is invalid.
        f. **system:** raises error.

- **Use-Case: Adding a product to the store basket 2.2.3-d — Failed Scenario(store identifier is invalid)**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
        a. product identifier.
        b. amount of product.
        c. store identifier.
    4. Postconditions:
    5. Expected result:

6. Actions:
    a. **user**: Requests to add a product with product identifier to shopping basket of store with store identifier.
    b. **system**: Validates store identifier.
    c. **system**: Finds out that store identifier is invalid
    d. **system**: raises error.

- **Use-Case: Adding a product to the store basket 2.2.3-e — Failed Scenario(product identifier is invalid)**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
        a. product identifier.
        b. amount of product.
        c. store identifier.
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Requests to add a product with product identifier to shopping basket of store with store identifier.
        b. **system**: Validates store identifier
        c. **system**: Validates product identifier.
        d. **system**: Finds out that product identifier is invalid
        e. **system:** raises error.

- **Use-Case: Observing the shopping cart 2.2.4.1-a — Success Scenario**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:

4. Postconditions:
5. Expected result: Relevant information regarding the user's shopping cart.
6. Actions:
    a. **guest**: Request to observe the shopping cart.
    b. **system**: Show relevant information regarding the user's shopping cart.

- **Use-Case: Observing the shopping cart 2.2.4.1-b — Failed Scenario (Shopping cart is empty)**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **guest**: Request to observe the shopping cart.
        b. **system**: Finds out the shopping cart is empty.
        c. **system:** sends an appropriate message to the user.

- **Use-Case: Removing a product from the shopping cart 2.2.4.2-a — Success Scenario**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
        a. product identifier.
        b. amount to remove
    4. Postconditions: for the shopping cart:

        num_of_product in cart_after == num_of_product_in_cart_before - amount to remove

5. Expected result:
6. Actions:
    a. **user**: Requests to remove the given amount of products with the given product identifier from his cart.
    b. **system**: Validates the product identifier.
    c. **system**: Validates the amount to remove.
    d. **system**: Reduce the specified amount of products with the given product identifier from the user's cart.

- **Use-Case: Removing a product from the shopping cart 2.2.4.3-a — Failed Scenario (product identifier is not valid)**
    1. Actor: user
    2. Preconditions: user is connected to the system.
    3. Parameters:
        a. product identifier.
        b. amount to remove.
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Requests to remove the given amount of products with the given product identifier from his cart.
        b. **system**: Validates the product identifier.
        c. **system**: Finds out product identifier is not valid
        d. **system:** raises an error.

- **Use-Case: Removing a product from the shopping cart 2.2.4.4-a — Failed Scenario (amount to remove is not valid)**
    1. Actor: user
    2. Preconditions: user is connected to the system.

3. Parameters:
    a. product identifier.
    b. amount to remove.
4. Postconditions:
5. Expected result:
6. Actions:
    a. **user**: Requests to remove the given amount of products with the given product identifier from his cart.
    b. **system**: Validates the product identifier.
    c. **system**: Validates the amount to remove.
    d. **system**: Finds out that the amount to remove is not valid
    e. **system:** Raises an error.

- **Use-Case: Payment 2.2.5-a — Success Scenario**
    1. Actor: user
    2. Preconditions:
        a. user is connected to the system.
        b. user has a non-empty shopping cart.
    3. Parameters:
        a. Payment method.
        b. Shopping cart.
        c. Payment details.
    4. Postconditions:
        a. for each store in which a product was purchased: num_of_product_before - num_purchased == num_of_product_after.
        b. The user is charged with the order's price according to the discount policy & purchase policy.
        c. Purchase info is logged in the system.
    5. Expected result: Confirmation that the purchase was successful.

6. Actions:
    a. **user**: Request to make a purchase for their shopping cart.
    b. **system**: Go over all shopping baskets in the user's shopping cart. for every shopping basket:
        i. **system**: Checks if the store associated with the shopping basket exists in the system
        ii. **system**: Checks if the store is open. If so, for every product in shopping basket:
            ★ **system**: Checks if the specified amount of the product is available at the store.
        iii. **system**: check that the store purchase policy is met.
    b. **system**: Requests for desired supply method.
    c. **user**: Enters desired supply method.
    d. **system**: Send the supply method and user's address to the external supply service.
    e. **system**: Request from the user for a payment method & payment details.
    f. **user**: Enters desired payment method & payment details.
    g. **system**: Send the payment method & payment details to the external payment service.
    h. **system**: Send the relevant package details & user details to the external delivery service.
    i. **system:** Updates store's supply.
    j. **system**: Logs purchase details in the system.
    k. **system**: Send confirmation to the user.

- **Use-Case: Payment 2.2.5-b — Failure Scenario (One of the stores is closed \ does not exist in the system)**
    1. Actor: user

2. Preconditions:
    a. user is connected to the system.
    b. user has a non-empty shopping cart.
3. Parameters:
    a. Payment method.
    b. Shopping cart.
    c. Payment details.
4. Postconditions:
5. Expected result:
6. Actions:
    a. **user**: Request to make a purchase for their shopping cart.
    b. **system**: Go over all shopping baskets in the user's shopping cart.
       for every shopping basket:
        ★ **system**: Checks if the store associated with the shopping basket is open.
    c. **system:** Find a closed store
    d. **system:** Raise an error to the user

- **Use-Case: Payment 2.2.5-c — Failure Scenario (Insufficient amount of product)**
    1. Actor: user
    2. Preconditions:
        a. user is connected to the system.
        b. user has a non-empty shopping cart.
    3. Parameters:
        a. Payment method.
        b. Shopping cart.
        c. Payment details.
    4. Postconditions:

5. Expected result:
6. Actions:
    a. **user**: Request to make a purchase for their shopping cart.
    b. **system**: Go over all shopping baskets in the user's shopping cart. for every shopping basket:
        i. **system**: Checks if the store associated with the shopping basket exists in the system.
        ii. **system:** Checks if the store associated is open.
        iii. If so, for every product in shopping basket:
            ★ **system**: Checks if the specified amount of the product is available at the store.
    c. **system**: Raise an error to the user

- **Use-Case: Payment 2.2.5-d — Failure Scenario (Store purchase policy is not met)**
    1. Actor: user
    2. Preconditions:
        a. user is connected to the system.
        b. user has a non-empty shopping cart.
    3. Parameters:
        a. Payment method.
        b. Shopping cart.
        c. Payment details.
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Request to make a purchase for their shopping cart.
        b. **system**: Go over all shopping baskets in the user's shopping cart. for every shopping basket:

i. **system**: Checks if the store associated with the shopping basket exists in the system

ii. **system**: Checks if the store is open. If so, for every product in shopping basket:

★ **system**: Checks if the specified amount of the product is available at the store.

iii. **system**: check that the store purchase policy is met.

c. **system**: Raise an error to the user since there exists a store for which its purchase policy isn't met.

- **Use-Case: Payment 2.2.5-e — Failure Scenario (Supply method not supported by external service / not supported to user's address)**
    1. Actor: user
    2. Preconditions:
        a. user is connected to the system.
        b. user has a non-empty shopping cart.
    3. Parameters:
        a. Payment method.
        b. Shopping cart.
        c. Payment details.
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Request to make a purchase for their shopping cart.
        b. **system**: Go over all shopping baskets in the user's shopping cart. for every shopping basket:
            i. **system**: Checks if the store associated with the shopping basket exists in the system.

ii. **system**: Checks if the store is open. If so, for every product in shopping basket:
    1. **system**: Checks if the specified amount of the product is available at the store.
iii. **system**: check that the store purchase policy is met.

c. **system**: Requests for desired supply method.
d. **user**: Enters desired supply method.
e. **system**: Send the supply method and user's address to the external supply service.
f. **system**: Raises an appropriate error stating that the selected supply method is not supported / not supported for his address.

- **Use-Case: Payment 2.2.5-f — Failure Scenario (External payment service fail)**
    1. Actor: user
    2. Preconditions:
        a. user is connected to the system.
        b. user has a non-empty shopping cart.
    3. Parameters:
        a. Payment method.
        b. Shopping cart.
        c. Payment details.
    4. Postconditions:
    5. Expected result:
    6. Actions:
        a. **user**: Request to make a purchase for their shopping cart.
        b. **system**: Go over all shopping baskets in the user's shopping cart. for every shopping basket:
            i. **system**: Checks if the store associated with the shopping basket exists in the system.

ii. **system**: Checks if the store is open. If so, for every product in shopping basket:
    1. **system**: Checks if the specified amount of the product is available at the store.
iii. **system**: check that the store purchase policy is met.

c. **system**: Requests for desired supply method.
d. **user**: Enters desired supply method.
e. **system**: Send the supply method and user's address to the external supply service.
f. **system**: Request from the user for a payment method & payment details.
g. **user**: Enters desired payment method & payment details.
h. **system**: Validates the payment method & Sends payment details to the external payment service.
i. **system**: Raise an error to the user stating that the payment failed with an appropriate specification.

- **Use-Case: logout 2.3.1-a — Success Scenario**
    1. Actor: member.
    2. Preconditions: member is logged in the system.
    3. Parameters:
    4. Postconditions:
        a. member shopping cart has been saved.
        b. user is now a guest and has an empty shopping cart.
    5. Actions:
        a. **member**: Requests from the system to logout.
        b. **System:** Saves member's shopping cart.
        c. **System:** Logs the member out.

- **Use-Case: Opening Store 2.3.2-a — Success Scenario**
    1. Actor: member.
    2. Preconditions: member is logged in.
    3. Parameters: store credentials.
    4. Postconditions: new store is created, and member is its owner.
    5. Actions:
        a. **member**: Requests the system to open a new store.
        b. **system**: Requests the member to enter store credentials.
        c. **member**: Enters store credentials.
        d. **system**: Validates store credentials.
        e. **system**: Creates a new store with the given store credentials
        f. **system:** Saves it into the system.
        g. **system**: Defines the member to be the store's owner.

- **Use-Case: Opening Store 2.3.2-b — Failed Scenario (store credentials are not valid)**
    6. Actor: member.
    7. Preconditions: member is logged in to the system.
    8. Parameters: store credentials.
    9. Postconditions:
    10. Actions:
        a. **member**: Requests the system to open a new store.
        b. **system**: Requests the member to enter store credentials.
        c. **member**: Enters store credentials.
        d. **system**: Validates store credentials.
        e. **system**: Find out that the store credentials are invalid
        f. **system:** Raises an error.

- **Use-Case: storage management- add product 2.4.1-a — Success Scenario**
    1. Actor: store owner | store manager
    2. Preconditions:
        a. store owner | store manager is logged in to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permission to add products to the store with the given store identifier.
    3. Parameters:
        a. product credentials.
        b. amount of product.
        c. store identifier.
    4. Postconditions: The product was added to the store with the given store identifier.
    5. Actions:
        a. **store owner | store manager**: Request the system to add a product to store with the given store identifier a product.
        b. **system**: Requests the store owner to enter product credentials.
        c. **store owner | store manager**: Enters product credentials.
        d. **system**: Validates product credentials.
        e. **system**: Requests for the desired amount of the product.
        f. **store owner | store manager**: Enters the desired amount of the product.
        g. **system**: Validates the entered amount.
        h. **system**: Adds the amount of product according to the product credentials to the store with the given store identifier.

- **Use-Case: storage management- add product 2.4.1-b — Failed Scenario (product credentials are not valid)**
    1. Actor: store owner | store manager
    2. Preconditions:
        a. store owner | store manager is logged in to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permission to add products to the store with the given store identifier.
    3. Parameters:
        a. product credentials.
        b. amount of product.
        c. store identifier.
    4. Postconditions:
    5. Actions:
        a. **store owner | store manager**: Request the system to add a product to store with the given store identifier a product.
        b. **system**: Requests the store owner to enter product credentials.
        c. **store owner | store manager**: Enters product credentials.
        d. **system**: Validates product credentials.
        e. **system**: Finds out product credentials are invalid
        f. **system:** Raises an error.

- **Use-Case: storage management- add product 2.4.1-c — Failed Scenario (amount of product is not valid)**
    1. Actor: store owner | store manager
    2. Preconditions:
        a. store owner | store manager is logged in to the system.

b. store with the given store identifier exists and is open.
c. store owner | store manager owns | manages the store with the given identifier.
d. store manager has permission to add products to the store with the given store identifier.

3. Parameters:
    a. product credentials.
    b. amount of product.
    c. store identifier.
4. Postconditions:
5. Actions:
    a. **store owner | store manager**: Request the system to add a product to store with the given store identifier a product.
    b. **system**: Requests the store owner to enter product credentials.
    c. **store owner | store manager**: Enters product credentials.
    d. **system**: Validates product credentials.
    e. **system**: Requests for the desired amount of the product.
    f. **store owner | store manager**: Enters the desired amount of the product.
    g. **system**: Validates the entered amount.
    h. **system**: Finds out that the amount of product is not valid
    i. **system:** Raises an error.

- **Use-Case: changing store purchase policy 2.4.2.1-a — Success Scenario**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.

c. store owner | store manager owns | manages the store with the given identifier.
d. store manager has permission to change the store's purchase policy of the store with the given store identifier.

3. Parameters:
    a. store identifier.
    b. purchase policy information.
4. Postconditions: purchase policy is defined by the given information.
5. Actions:
    a. **store owner | store manager**: Requests the system to change purchase policy in store with the given store identifier.
    b. **system**: Prompts the store owner to enter purchase policy information.
    c. **store owner | store manager**: Enters purchase policy information.
    d. **system**: Validates purchase policy information.
    e. **system**: Replace the existing purchase policy at the store with the given store identifier with a new purchase policy which is defined according to the given information.

- **Use-Case: changing store purchase policy 2.4.2.1-b — Failed Scenario (purchase policy information is not valid)**
    1. Actor: store owner.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.

    d. store manager has permission to change the store's purchase policy of the store with the given store identifier.
3. Parameters:
    a. store identifier.
    b. purchase policy information.
4. Postconditions:
5. Actions:
    a. **store owner | store manager**: Asks system to change purchase policy in store with the given store identifier.
    b. **system**: Prompt user to enter purchase policy information.
    c. **store owner | store manager**: Enter purchase policy information.
    d. **system**: Validates purchase policy information.
    e. **system**: Finds out that the purchase policy information is not valid
    f. **system:** Raises an error.

- **Use-Case: changing store purchase types 2.4.2.2-a — Success Scenario**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permission to change the store's purchase types of the store with the given store identifier.
    3. Parameters:
        a. store identifier.
        b. purchase types information.

4. Postconditions: purchase types in the store with the given store identifier are defined by the given types.
5. Actions:
    a. **store owner | store manager**: Requests the system to change purchase types in store with the given store identifier.
    b. **system**: Prompts the store owner to enter purchase types information.
    c. **store owner | store manager**: Enters purchase types information.
    d. **system**: Validates the given purchase types information.
    e. **system**: Replace the purchase types in the store with the given store identifier with purchase types which are defined by the given information.

- **Use-Case: changing store purchase types 2.4.2.2-b — Failed Scenario (purchase types information is not valid)**
    1. Actor: store owner.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permission to change the store's purchase types of the store with the given store identifier.
    3. Parameters:
        a. store identifier.
        b. purchase types information.
    4. Postconditions:
    5. Actions:

a. **store owner | store manager**: Requests the system to change purchase types in store with the given store identifier.
b. **system**: Prompts the store owner to enter purchase types information.
c. **store owner | store manager**: Enters purchase types information.
d. **system**: Validates the given purchase types information.
e. **system**: Finds out that the given purchase types information is not valid.
f. **system**: Raises an error.

- **Use-Case: changing store discount types 2.4.2.3-a — Success Scenario**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permission to change the store's discount types of the store with the given store identifier.
    3. Parameters:
        a. store identifier.
        b. discount types information.
    4. Postconditions: discount types in the store with the given store identifier are defined by the given types.
    5. Actions:
        a. **store owner | store manager**: Requests the system to change discount types in store with the given store identifier.

b. **system**: Prompts the store owner to enter discount types information.
c. **store owner | store manager**: Enters discount types information.
d. **system**: Validates the given discount types information.
e. **system**: Replace the discount types in the store with the given store identifier with discount types which are defined by the given information.

- **Use-Case: changing store discount types 2.4.2.3-b — Failed Scenario (discount types information is not valid)**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permission to change the store's discount types of the store with the given store identifier.
    3. Parameters:
        a. store identifier.
        b. discount types information.
    4. Postconditions:
    5. Actions:
        a. **store owner | store manager**: Requests the system to change discount types in store with the given store identifier.
        b. **system**: Prompts the store owner to enter discount types information.
        c. **store owner | store manager**: Enters discount types information.
        d. **system**: Validates the given discount types information.

e. **system**: Finds out that the given discount types information is not valid.
f. **system:** Raises an error.

- **Use-Case: changing store discount policy 2.4.2.4-a — Success Scenario**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permission to change the store's discount policy of the store with the given store identifier.
    3. Parameters:
        a. store credentials.
        b. discount policy information.
    4. Postconditions: discount policy is defined by the given information.
    5. Actions:
        a. **store owner | store manager**: Requests the system to change the discount policy in store with the given store identifier.
        b. **system**: Prompts the store owner to enter discount policy information.
        c. **store owner | store manager**: Enters discount policy information.
        d. **system**: Validates discount policy information.
        e. **system**: Replace the existing discount policy at the store with the given store identifier with a new discount policy which is defined according to the given information.

- **Use-Case: changing store discount policy 2.4.2.4-b — Failed Scenario (discount policy information is not valid)**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permission to change the store's discount policy of the store with the given store identifier.
    3. Parameters:
        a. store credentials.
        b. discount policy information.
    4. Postconditions:
    5. Actions:
        a. **store owner | store manager**: Requests the system to change the discount policy in store with the given store identifier.
        b. **system**: Prompts the store owner to enter discount policy information.
        c. **store owner | store manager**: Enters discount policy information.
        d. **system**: Validates discount policy information.
        e. **system**: Finds out that the given discount policy information is not valid.
        f. **system:** raises an error.

- **Use-Case: appointing another store owner 2.4.3-a — Success Scenario**
    1. Actor: store owner.
    2. Preconditions:

a. store owner is connected to the system.
b. store with the given store identifier exists and is open.
c. store owner owns the store with the given identifier.

3. Parameters:
    a. store identifier.
    b. member identifier.
4. Postconditions: store owner promotion proposal is sent to the member with the given member identifier.
5. Actions:
    a. **store owner**: Requests from the system to add an additional store owner to the store with the given store identifier.
    b. **system**: Prompts store owner to enter member identifier.
    c. **store owner**: Enters member's identifier.
    d. **system**: Validates member identifier.
    e. **system**: Verifies that he is not already a store owner of the store with the given store identifier.
    f. **system**: Sends notification to member with the given member identifier, asking if they agree to the promotion.

- **Use-Case: appointing a store owner 2.4.3-b — Failed Scenario (invalid member credentials)**
    1. Actor: store owner.
    2. Preconditions:
        a. store owner is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner owns the store with the given identifier.
    3. Parameters:
        a. store identifier.
        b. member identifier.

4. Postconditions:
5. Actions:
    a. **store owner**: Requests from the system to add an additional store owner to the store with the given store identifier.
    b. **system**: Prompts store owner to enter member identifier.
    c. **store owner**: Enters member's identifier.
    d. **system**: Validates member identifier.
    e. **system**: Finds out member identifier is not valid
    f. **system:** Raises error.

- **Use-Case: appointing another store owner 2.4.3-c — Failed Scenario (member with the given identifier is already store owner)**
    1. Actor: store owner.
    2. Preconditions:
        a. store owner is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner owns the store with the given identifier.
    3. Parameters:
        a. store identifier.
        b. member identifier.
    4. Postconditions:
    5. Actions:
        a. **store owner**: Requests from the system to add an additional store owner to the store with the given store identifier.
        b. **system**: Prompts store owner to enter member identifier.
        c. **store owner**: Enters member's identifier.
        d. **system**: Validates member identifier.
        e. **system:** Verifies that he is not already a store owner of the store with the given store identifier.

f. **system**: Finds out that the member with the given member identifier is already a store owner of the store with the given store identifier.
g. **system:** Raises an error.

- **Use-Case: accepting promotion to store owner 2.4.3-a — Success Scenario**
    1. Actor: member
    2. Preconditions:
        a. member is logged in the system.
        b. member got offered by a store owner with the given store owner identifier to be store owner of store with given store identifier.
        c. member is not a store owner of the store with the given store identifier.
        d. store with the given store identifier exists and is open.
        e. store owner with the given store identifier owns the store with the given store identifier.
    3. Parameters:
        a. store owner promotion offer notification information.
        b. store identifier.
        c. store owner identifier.
    4. Postconditions:
        a. member is now a store owner of the store with the given store identifier.
        b. store owner with the given store owner identifier is the one who hired the member.
        c. store owner with the given store owner identifier got sent a notification stating the member accepted the promotion.
    5. Actions:

a. **system:** Renders member store owner promotion offer notification information.
b. **member**: accepts the promotion.
c. **system**: Assigns the member the role of a store owner of the store with the given store identifier.
d. **system**: Assigns the store owner with the given store owner identifier to be the one who hired member to be store owner of the store with the given store identifier.
e. **system**: Sends a notification to the store owner with the given store owner identifier stating that the member accepted the promotion.

- **Use-Case: accepting promotion to store owner 2.4.3-b — Failed Scenario (member rejects offer)**
    1. Actor: member
    2. Preconditions:
        a. member is logged in the system.
        b. member got offered by a store owner with the given store owner identifier to be store owner of store with given store identifier.
        c. member is not a store owner of the store with the given store identifier.
        d. store with the given store identifier exists and is open.
        e. store owner with the given store identifier owns the store with the given store identifier.
    3. Parameters:
        a. store owner promotion offer notification information.
        b. store identifier.
        c. store owner identifier.

4. Postconditions: the store owner with the given store owner identifier got sent a notification stating the member rejected the promotion.
5. Actions:
    a. **system:** Renders member store owner promotions offer notification information.
    b. **member**: Rejects the promotion.
    c. **system**: Sends the store owner with the given store owner identifier a notification stating that the member rejected the promotion.

- **Use-Case: appointing a store manager 2.4.6-a — Success Scenario**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permissions of adding store managers to the store with the given store identifier.
    3. Parameters:
        a. store identifier.
        b. member identifier.
        c. manager permissions.
    4. Postconditions: manager owner promotion proposal is sent to the member with the given member identifier.
    5. Actions:
        a. **store owner | store manager**: Requests from the system to add an additional store manager to the store with the given store identifier.

b. **system**: Prompts store owner to enter member identifier.
c. **store owner | store manager**: Enters member's identifier.
d. **system**: Validates member identifier.
e. **system:** Verifies that he is not already a store owner / store manager of the store with the given store identifier.
f. **system**: Requests for manager permissions.
g. **store owner | store manager**: Enters manager permissions.
h. **system**: Validates manager permissions.
i. **system**: Sends notification to the member with the given member identifier, asking if they agree to the promotion.

- **Use-Case: appointing a store manager 2.4.6-b — Failed Scenario (invalid member credentials)**
    1. Actor: store owner.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permissions of adding store managers to the store with the given store identifier.
    3. Parameters:
        a. store identifier.
        b. member identifier.
        c. manager permissions.
    4. Postconditions:
    5. Actions:

a. **store owner | store manager**: Requests from the system to add an additional store manager to the store with the given store identifier.
b. **system**: Prompts store owner to enter member identifier.
c. **store owner | store manager**: Enters member's identifier.
d. **system**: Validates member identifier.
e. **system**: Finds out that the given member identifier is not valid
f. **system:** Raises error.

- **Use-Case: appointing a store manager 2.4.6-c — Failed Scenario (member is already store owner / store manager)**
    1. Actor: store owner.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permissions of adding store managers to the store with the given store identifier.
    3. Parameters:
        a. store identifier.
        b. member identifier.
        c. manager permissions.
    4. Postconditions:
    5. Actions:
        a. **store owner | store manager**: Requests from the system to add an additional store manager to the store with the given store identifier.
        b. **system**: Prompts store owner to enter member identifier.

c. **store owner | store manager**: Enters member's identifier.
d. **system**: Validates member identifier.
e. **system:** Verifies that he is not already a store owner / store manager of the store with the given store identifier.
f. **system**: Finds out that the member with the given member identifier is already a store owner / store manager of the store with the given store identifier
g. **system:** Raises an error.

- **Use-Case: appointing a store manager 2.4.6-d — Failed Scenario (manager permissions are not valid)**
    1. Actor: store owner.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager owns | manages the store with the given identifier.
        d. store manager has permissions of adding store managers to the store with the given store identifier.
    3. Parameters:
        a. store identifier.
        b. member identifier.
        c. manager permissions.
    4. Postconditions:
    5. Actions:
        a. **store owner | store manager**: Requests from the system to add an additional store manager to the store with the given store identifier.
        b. **system**: Prompts store owner to enter member identifier.

c. **store owner | store manager**: Enters member's identifier.
d. **system**: Validates member identifier.
e. **system:** Verifies that he is not already a store owner / store manager of the store with the given store identifier.
f. **system**: Requests for manager permissions.
g. **store owner | store manager**: Enters manager permissions.
j. **system**: Validates manager permissions.
k. **system**: Finds out that the given manager permissions are not valid.
l. **system:** Raises an error.

- **Use-Case: accepting promotion to store manager 2.4.6-a — Success Scenario**
    1. Actor: member
    2. Preconditions:
        a. member is logged in the system.
        b. member got offered by a store owne | store managerr with the identifier to be store manager of store with given store identifier and have the given permissions.
        c. the given manager permissions are valid.
        d. member is not a store owner | store manager of the store with the given store identifier.
        e. store with the given store identifier exists and is open.
        f. store owner | store manager with the given identifier owns | manages the store with the given store identifier.
    3. Parameters:
        a. store manager promotion offer notification information.
        b. store identifier.
        c. store owner | store manager identifier (will be referred as "identifier").

    d. manager permissions.
4. Postconditions:
    a. member is now a store manager of the store with the given store identifier, and has the given permissions.
    b. store owner | store manager with the given identifier is the one who hired the member.
    c. store owner | store manager with the given identifier got sent a notification stating the member accepted the promotion.
5. Actions:
    a. **system:** Renders member store manager promotion offer notification information.
    b. **member**: Accepts the promotion.
    c. **system**: Assigns the member the role of a store manager of the store with the given store identifier with the given manager permissions.
    d. **system**: Assigns the store owner | store manager with the given identifier to be the one who hired member to be store manager of the store with the given store identifier.
    e. **system**: Sends a notification to the store owner | store manager with the given identifier stating that the member accepted the promotion.

- **Use-Case: accepting promotion to store manager 2.4.6-b — Failed Scenario (member rejects the offer)**
    1. Actor: member
    2. Preconditions:
        a. member is logged in the system.

b. member got offered by a store owne | store manager with the identifier to be store manager of store with given store identifier and have the given permissions.
c. the given manager permissions are valid.
d. member is not a store owner | store manager of the store with the given store identifier.
e. store with the given store identifier exists and is open.
f. store owner | store manager with the given identifier owns | manages the store with the given store identifier.

3. Parameters:
    a. store manager promotion offer notification information.
    b. store identifier.
    c. store owner | store manager identifier (will be referred as “identifier”).
    d. manager permissions.
4. Postconditions: store owner | store manager with the given identifier got sent a notification stating the member rejected the promotion.
5. Actions:
    a. **system:** Renders member store manager promotion offer notification information.
    b. **member**: Rejects the promotion.
    c. **system**: Sends a notification to the store owner | store manager with the given identifier stating that the member rejected the promotion.

- **Use-Case: change store manager permissions 2.4.7-a — Success Scenario**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.

b. store with the given store identifier exists and is open.
c. store owner | store manager is an owner | manager of the store with the given store identifier.
d. store manager has permission to change store manager permissions to the store managers he hired for store with the given store identifier.

3. Parameters:
    a. store manager identifier.
    b. store identifier.
    c. manager permissions.
4. Postconditions: store manager permissions were changed to be the given manager permissions.
5. Actions:
    a. **store owner | store manager**: Requests to change manager permissions for manager of store with store identifier.
    b. **system**: Requests for the store manager identifier.
    c. **store owner**: Enters store manager identifier.
    d. **system**: Validates store manager identifier.
    e. **system**: Validate that the store owner | store manager hired the store manager with the given store manager identifier to the store with the given store identifier.
    f. **system**: Requests for new manager permissions.
    g. **store owner | store manager**: Enters manager permissions.
    h. **system**: Validates manager permissions.
    i. **system**: Updates the manager permissions of store manager with the given store manager identifier at store with the given store identifier to be the given manager permissions.

- **Use-Case: change store manager permissions 2.4.7-b — Failed Scenario (store manager identifier not valid)**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager is an owner | manager of the store with the given store identifier.
        d. store manager has permission to change store manager permissions to the store managers he hired for the store with the given store identifier.
    3. Parameters:
        a. store manager identifier.
        b. store identifier.
        c. manager permissions.
    4. Postconditions:
    5. Actions:
        a. **store owner | store manager**: Requests to change manager permissions for manager of store with store identifier.
        b. **system**: Requests for the store manager identifier.
        c. **store owner**: Enters store manager identifier.
        d. **system**: Validates store manager identifier.
        e. **system**: Finds out that the given store manager identifier is not valid.
        f. **system:** Raises an error.

- **Use-Case: change store manager permissions 2.4.7-c — Failed Scenario (store owner | store manager didn’t hire the store manage)**
    1. Actor: store owner | store manager.

2. Preconditions:
    a. store owner | store manager is connected to the system.
    b. store with the given store identifier exists and is open.
    c. store owner | store manager is an owner | manager of the store with the given store identifier.
    d. store manager has permission to change store manager permissions to the store managers he hired for store with the given store identifier.
3. Parameters:
    a. store manager identifier.
    b. store identifier.
    c. manager permissions.
4. Postconditions:
5. Actions:
    a. **store owner | store manager**: Requests to change manager permissions for manager of store with store identifier.
    b. **system**: Requests for the store manager identifier.
    c. **store owner**: Enters store manager identifier.
    d. **system**: Validates store manager identifier.
    e. **system**: Validate that the store owner | store manager hired the store manager with the given store manager identifier to the store with the given store identifier.
    f. **system**: Finds out that the store manager with the given store identifier was not hired by the store owner | store manager that initiated the request.
    g. **system:** Raises an error.

- **Use-Case: closing a store 2.4.9-a — Success Scenario**
    1. Actor: store owner

2. Preconditions:
    a. store owner is connected to the system.
    b. store with the given store identifier exists and is open.
    c. store owner is the founder of the store with the given identifier.
3. Parameters: store identifier.
4. Postconditions:
    a. The store with the given store identifier is closed.
    b. All other store owners | store managers of the store with the given store identifier were sent a notification that the store is closed.
5. Actions:
    a. **store owner**: Requests the system to close the store with the given store identifier.
    b. **system**: Validates the store identifier.
    c. **system**: Sets the store with the given store identifier as closed.
    d. **system**: Notifies all store owner | store managers of the store with the given store identifier that the store is closed.

- **Use-Case: view information regarding store employees 2.4.11-a — Success Scenario**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager is an owner | manager of the store with the given identifier.
        d. store manager has permissions to view information regarding store employees.
    3. Parameters: store identifier.

4. Postconditions: —
5. Expected result: relevant information regarding the employees of the store with the given store identifier.
6. Actions:
    a. **store owner | store manager**: Requests the system for details about the store employees.
    b. **system**: Prompts the user to enter the store identifier.
    c. **store owner**: Enters the store identifier.
    d. **system**: Validates the given store identifier.
    e. **system**: Returns relevant information regarding the employees of the store with the given store identifier.

- **Use-Case: view information regarding store employees 2.4.11-b — Failed Scenario (store identifier is not valid)**
    1. Actor: store owner | store manager.
    2. Preconditions:
        a. store owner | store manager is connected to the system.
        b. store with the given store identifier exists and is open.
        c. store owner | store manager is an owner | manager of the store with the given identifier.
        d. store manager has permissions to view information regarding store employees.
    3. Parameters: store identifier.
    4. Postconditions: —
    5. Actions:
        a. **store owner | store manager**: Requests the system for details about the store employees.
        b. **system**: Prompts the user to enter the store identifier.
        c. **store owner**: Enters the store identifier.

d. **system**: Validates the given store identifier.
e. **system**: Finds out that the given store identifier is not valid .
f. **system:** Raises an error.

- **Use-Case: Getting information about the purchases history in a store**

  2.4.13-a — Success Scenario

  1. Actor: store owner | store manager.
  2. Preconditions:
     a. store owner | store manager is connected to the system.
     b. store with the given store identifier exists and is open.
     c. store owner | store manager is an owner | manager of the store with the given identifier.
     d. store manager has permissions to view information regarding store employees.
  3. Parameters: store identifier
  4. Postconditions: —
  5. Expected result: relevant details of all purchases of customers in the store.
  6. Actions:
     a. **store owner | store manager**: Requests system to view purchases history of the store with the given store identifier.
     b. **system**: Validates the store identifier.
     c. **system**: Returns the relevant information of purchases history for the store with the given store identifier.

- **Use-Case: closing a store 6.1**

  1. Actor: system manager.
  2. Preconditions:
     a. user is connected to the system.

b. store is open.

3. Parameters: store credentials.
4. Postconditions:
    a. the store is deleted.
    b. store employees are demoted.
5. Actions:
    a. **system manager**: Asks system to close a store.
    b. **system**: Prompts User to enter store credentials.
    c. **system manager**: Enters store credentials.
    d. **system**: Validates store credentials.
    e. **system**: Closes the store.
    f. **system**: Notifies the store employees and demotes them.

- **Use-Case: deleting a system member 6.2**
    1. Actor: system manager.
    2. Preconditions:
        a. user is connected to the system.
        b. The system manager is online.
    3. Parameters: member name.
    4. Postconditions:
        a. member is no longer registered to the system.
        b. if the member was a store owner, then every employee that member appointed will be demoted.
        c. if the member was a store founder, then their store is deleted and the employees are demoted.
    5. Actions:
        a. **system manager**: Asks to delete a member.
        b. **system**: Validates member name.
        c. **system**:

i. if member is store founder, delete the stores that were founded by them and demote their employees.
ii. else if member is a store owner, demote all employees appointed by the member.

d. **system**: delete member.

- **Use-Case: reading message inbox 6.3**
    1. Actor: system manager.
    2. Preconditions:
        a. user is connected to the system.
    3. Parameters: —
    4. Postconditions: —
    5. Actions:
        a. **system manager**: Asks the system to view the message inbox.
        b. **system**: Show system manager the message inbox.

- **Use-Case: sending messages to members 6.3**
    1. Actor: system manager.
    2. Preconditions:
        a. user is connected to the system.
    3. Parameters:
        a. recipients (member name).
        b. message text.
    4. Postconditions: message delivered.
    5. Actions:
        a. **system manager**: Asks the system to send a message.
        b. **system**: Prompt user to enter message and recipients.
        c. **system manager**: Enter required information.
        d. **system**: Validate members' names.

e. **system**: send messages.

- **Use-Case: Getting information about the purchases history of a member in the system 2.6.4-a — Success Scenario**
    1. Actor: system manager.
    2. Preconditions: system manager is connected to the system.
    3. Parameters: user identifier
    4. Postconditions: —
    5. Expected result: relevant details of all purchases of user with the given user identifier.
    6. Actions:
        a. **system manager**: Requests system for details regarding the purchase history of the user with the given user identifier.
        b. **system**: Verifies user identifier.
        c. **system**: Returns relevant details of all purchases of user with the given user identifier.

- **Use-Case: Getting information about the purchases history of a member in the system 2.6.4-b — Failed Scenario (user identifier is not valid)**
    1. Actor: system manager.
    2. Preconditions: system manager is connected to the system.
    3. Parameters: user identifier
    4. Postconditions: —
    5. Actions:
        a. **system manager**: Requests system for details regarding the purchase history of the user with the given user identifier.
        b. **system**: Verifies user identifier.
        c. **system**: Finds out that the given user identifier is not valid.
        d. **system:** Raises an error.

- **Use-Case: Getting information about the purchases history in a store 2.6.4-a — Success Scenario**
    1. Actor: system manager.
    2. Preconditions: system manager is connected to the system.
    3. Parameters: store identifier
    4. Postconditions: —
    5. Expected result: relevant details of all purchases in the store with the given store identifier.
    6. Actions:
        a. **system manager**: Requests system for details regarding the purchase history in the store with the given store identifier.
        b. **system**: Verifies store identifier.
        c. **system**: Returns relevant details of all purchases in the store with the given store identifier.

- **Use-Case: Getting information about the purchases history in a store in the system 2.6.4-b — Failed Scenario (store identifier is not valid)**
    1. Actor: system manager.
    2. Preconditions: system manager is connected to the system.
    3. Parameters: store identifier
    4. Postconditions: —
    5. Actions:
        a. **system manager**: Requests system for details regarding the purchase history in the store with the given store identifier.
        b. **system**: Verifies store identifier.
        c. **system**: Finds out that the given store identifier is not valid and raises an error.

- **Use-Case: Getting information about the system 6.5**
    1. Actor: system manager.
    2. Preconditions: user is connected to the system.
    3. Parameters: —
    4. Postconditions: —
    5. Expected result: predefined details about the system.
    6. Actions:
        a. **system manager**: Asks system for the wanted details.
        b. **system**: Returns the requested details.