Use Cases – תרחישי שימוש

System + System Administrator use cases:

Use case – 1: **System initialization**

1. **Actors:** System.

2. **Pre-conditions:** None

3. **Post-Conditions:** The system is up and running, and users can log in and use it.

4. **Input parameters:** System manager user information, different services information (how to connect to Visa, to Suppliers etc.)

5. **Actions:**

   a) The system asks for user information for the system manager.
   b) Authentication – The system checks if the user is subscribed, and if so, sets him as the system manager.
   c) The system connects to each service provider and checks if the service is up and ready to accept messages.
   d) If Everything is working the system moves to the main screen, and ready to handle different users and stores.

6. **Alternatives/Extensions:**

   a) If the user information are not correct, or the user given is not subscribed, the system asks again for details and not moving to the next screen.
   b) If one of the services fail to receive messages, the system will send notification to the system manager and will halt on the "waiting for the services screen".

7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The system manager try to login with a wrong user information.	Invaild user information, valid services information	Failed authentication, a matching error message will be returned.	Bad
The system manager try to conncect to the wrong services (payment, suppliers).	Valid user information, Invalid services information	Failed services connection, a matching error message will be returned.	Bad
The system manager was able to connect to the system and initialize the different services.	Valid user information, valid services information	Success, the system will initialize properly.	Good

<u>Use case – 2:</u> Change services

1. **Actors:** System.
2. **Pre-conditions:** The new service exists and is ready to receive messages.
3. **Post-Conditions:** The system can use the new service as usual.
4. **Input parameters:** Information about new service (id, type of service etc.)
5. **Actions:**
    a) The system connects to the new service and checks if it is ready.
    b) When the new service is ready the system removes the old service.
    c) The system sets the new service as the default service for the correct type (Pay, supply etc.)
6. **Alternatives/Extensions:**
    a) If the new service fails to start or the information is incorrect, the system will not let the users to remove the old service.
7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The system try to use a invalid service	Invaild information about the new service	Failed authentication, a matching error message will be returned.	Bad
The system try to change the specific service while the service it handles other transcations.	Valid information	Failed services update, a matching error message will be returned.	Bad
The sysem received correct data and will update the system.	Valid information	Success, the service will update properly.	Good

<u>Use case – 3:</u> Using the services (Pay, Supply)

1. **Actors:** System, Users.

2. **Pre-conditions:** The information that should be sent to the service was collected or created correctly.

3. **Post-Conditions:** The service accepted the request and did what it had to do successfully.

4. **Input parameters:** The parameters that should be sent to the service. (User information, card info, user address etc.)

5. **Actions:**

   a) The system validates that all the parameters are correct.
   b) The system calls the service using the correct API and passes all the parameters.
   c) If the received answer is positive, show a message to the user that the action was done successfully.

6. **Alternatives/Extensions:**

   a) If some of the arguments that were received from the user are incorrect, error message will be shown to the user, and he will be asked to fill this parameter again.
   b) If the service return a negative answer, handle it by the reason. (Wrong information – Correct it. Service too busy – Make the request later, etc.)

7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The service try to process an incorrect transaction.	Invaild transaction details	Failed authentication, a matching error message will be returned.	Bad
The service try to charge an invalid card details.	Valid transaction details, Invalid card info	Failed Charge failed, a matching error message will be returned.	Bad
The service was unable to process the payment.	Valid transaction details, valid card info	Failed Charge failed, a matching error message will be returned.	Bad
The payment service was able to charge the card succesfully.	Valid transaction details, Valid card info	Success, the payment was successfully received.	Good

Scenario Description	Parameters	Expected Result	Scenario
The service try to process an incorrect supply transaction.	Invaild transaction details	Failed authentication, a matching error message will be returned.	Bad
The service try to supply the wrong packages.	Valid transaction details, Invalid package information	Failed supply failed, a matching error message will be returned.	Bad
The service was unable to send a delivery confirmation receipt.	Valid transaction details, valid package information	Failed supply failed, a matching error message will be returned.	Bad
The supply service was able to process the pacakges info and to send a confirmation receipt.	Valid transaction details, Valid package information	Success, the supply was successfully received.	Good

<u>Use case – 4:</u> Pop up messages

1. **Actors:** System.

2. **Pre-conditions:** A subscribed user exists.

3. **Post-Conditions:** The user receives a pop-up message in real time or when he enters the system.

4. **Input parameters:** User phone number, the message

5. **Actions:**

    a) Check what is the type of action, and if the system has to send the message to his phone number or save it and show it to the user when he enters the system.
    b) According to the result of the check, send notification if needed, and save it in the correct database.
    c) When the user enters the system, check if there are any new messages that need to be shown to him.

6. **Alternatives/Extensions:**

    a) If the system is unable to send pop-up message, ask from the user for the correct phone number or permissions.

7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The system try to send a message to the wrong user.	Invaild user information	Failed authentication, a matching error message will be returned.	Bad
The system try to send a wrong message	Valid user information, Invalid message	Failed to pop up massages, a matching error message will be returned.	Bad
The system received corrcet data and will send it to the given user.	Valid user information, Valid message	Success, the system will send the message to the user properly.	Good

<u>Use case – 5:</u> Market history

1. **Actors:** User who are system manager.

2. **Pre-conditions:** Market working and there are stores and buyers.

3. **Post-Conditions:** The System manager has all the information about the stores and user's activities.

4. **Input parameters:** Store information, or the buyer information. (id, name, etc.)

5. **Actions:**

    a) The system manager writes as the input the ID of the store or buyer that he wants to check the history of.

    b) If the ID is correct, the system accesses the database and assembles a new file which contains all the information about the correct object. (Store, User…)

    c) The system returns the file with the relevant data to the store manager.

6. **Alternatives/Extensions:**

    a) If the information is incorrect, the system asks for the correct information again.
    b) If the system fails to assemble the file, it shows proper error message.

7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The manager try to see the purchases history of an invalid user.	Invaild information	Failed authentication, a matching error message will be returned.	Bad
The system manager requests to see the purchases history and at the same time another purchase is made in the same account, so he will not receive an updated histoty.	Valid information	Failed to receive the updated history, a matching error message will be returned.	Bad
The system received corrcet information and the manager is able to see the purchases history.	Valid information	Success, the manager will see the history that he wanted.	Good

Store Visitor use cases (subscriber, guest):

Use case – add searched product to cart

1. **Actors:** User
2. **Pre-conditions:**
    1. none
3. **Post-conditions:**
    - The product is added to the user cart
4. **Input Parameters:**
    1. User token
    2. Search string params (key words)
    3. Product filters string params (price range, ratings, category, name)
    4. Store filters string params (ratings, name, id)
    5. quantity
5. **Main Scenario:**
    1. The user initiates add to cart process with his token and search params
    2. The system authenticates the user
    3. The product is being added to the cart
6. **Alternatives**:
    1. If the user is not authenticated, a matching error message will be returned and the product wont be added
    2. If the product search params are incorrect, a matching error message will be returned and the product wont be added

Test 1: Successful addition to cart

- **Parameters:**
    - Valid token
    - Valid Search string params
    - Valid Product filters string params
    - Valid Store filters string params
    - Valid Quantity
- **Expected Result:**
    - The product is successfully added to the user's cart.

Test 2: User authentication failure

- **Parameters:**
    - Invalid token
    - Valid Search string params
    - Valid Product filters string params
    - Valid Store filters string params
    - Valid Quantity
- **Expected Result:**
    - Error message
    - The product is not added to the cart.

Test 3: Incorrect product search parameters

- **Parameters:**
    - Valid token
    - Invalid Search string params:
    - Valid Product filters string params
    - Valid Store filters string params
    - Valid Quantity
- **Expected Result:**
    - Error message
    - The product is not added to the cart.

Use case - 2: purchase process

1) **Actors:** User
2) **Pre-conditions:**
    a. User already added at least 1 product to cart
    b. Order is valid (each product quantity is in stock)
3) **Post-conditions:**
    a. **Success:**
        i. All purchased items are deducted from store's inventory.
        ii. Payment from the customer to the shop's owner is successfully processed.
    b. **Failure:**
        i. System reverts to pre-purchase state (atomic action).
4) **Input Parameters:**
    a. User payment parameters (card number, card date, card cvv)
    b. Address of the user
5) **Actions:**
    a. User initiates purchase order provides payment information and address to complete the purchase.
    b. The system authenticates the user
    c. The system checks the availability of the products in the cart
    d. The system charges the user with his payment information
    e. The system ordering a delivery for the items in the cart to the given address.
6) **Alternatives:**
    a. If the user is not authenticated, a matching error message will be returned, the purchase will not be completed, The system reverts to the pre-purchase state, the user wont be charged
    b. if there is an unavailable product in the cart, a matching error message will be returned, the purchase will not be completed, The system reverts to the pre-purchase state, the user wont be charged
    c. if the payment fails, a matching error message will be returned, the purchase will not be completed, The system reverts to the pre-purchase state, the user wont be charged

d. if the delivery fails, a matching error message will be returned, the purchase will not be completed, The system reverts to the pre-purchase state, the user wont be charged

Test 1: Successful purchase

- **Parameters:**
  - User token: valid token
  - User payment parameters: valid user payment information
  - Address of the user: valid address
- **Expected Result:**
  - Products are deducted from inventory.
  - Payment is processed successfully.
  - Delivery is ordered to the given address.

User authentication failure

- **Parameters:**
  - User token: invalid token
  - User payment parameters: valid user payment information
  - Address of the user: valid address
- **Expected Result:**
  - Error message
  - The purchase is not completed.
  - The system reverts to the pre-purchase state.

Unavailable product in cart

- **Parameters:**
  - User token: valid token
  - User payment parameters: valid user payment information
  - Address of the user: valid address
  - Cart: includes unavailable product
- **Expected Result:**
  - Error message
  - The purchase is not completed.

- The system reverts to the pre-purchase state.

Payment failure

- **Parameters:**
  - User token: valid token
  - User payment parameters: invalid payment information
  - Address of the user: valid address
- **Expected Result:**
  - Error message
  - The purchase is not completed.
  - The system reverts to the pre-purchase state.

Delivery failure

- **Parameters:**
  - User token: valid token
  - User payment parameters: valid user payment information
  - Address of the user: invalid address
- **Expected Result:**
  - Error message
  - The purchase is not completed.
  - The system reverts to the pre-purchase state.

Use-Case 3: Subscriber shop opening process

1. **Actors:** Subscriber (Founder)

2. **Pre-conditions:**
    - Subscriber is logged in to the system.

3. **Post-conditions:**
    - New store created.
    - Subscriber assigned as founder.

4. **Input Parameters:**
    - User token
    - Store name
    - contact information: email, phoneNumber, address
    - store description
    - Inventory

5. **Actions:**
    a) Subscriber initiates the process to open their store.
    b) Subscriber provides store name, description, contact information.
    c) Program creates a new store, and turn Subscriber into a Founder
    d) Subscriber completes the shop opening process and now owns a shop

6. **Scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
Outdated Token	User credentials	an error message pops stating "User's credentials unavailable"	Alternative
Shop name is taken	Store name	an error message pops stating "shop name is already taken"	Alternative
User opens a store	Store name, contact information, store description	Success, a shop was created for the given user	Happy

Use Case: Login

Actors:

- User

Preconditions:

- The user is registered in the system and has valid login credentials.

Postconditions:

1. The user is authenticated and logged in as a Visitor-Subscriber.
2. A user session is created, and the user can access subscriber-specific features.

Input Parameters:

- Username
- Password

Main Scenario:

1. User tries to login with username and password
2. The system verifies the provided credentials against the registered user database.
3. The user gets a token and loges in.

Extensions/Alternatives:

1. If the credentials are invalid, the system displays an error message indicating the failure.
2. If the account is suspended, a matching error

Test 1: Successful login

- **Preconditions:** The user is registered in the system and has valid login credentials.
- **Steps:**
    1. User enters the username and password.
    2. The system verifies the credentials against the registered user database.
    3. The system generates a token and logs in the user.
- **Expected Result:**
    - User is authenticated.
    - User is logged in.

Test 2: Unsuccessful login due to invalid credentials

- **Preconditions:** The user is registered in the system but provides invalid login credentials.
- **Steps:**
    1. User enters the username and incorrect password.
    2. The system verifies the credentials against the registered user database.
- **Expected Result:**
    - Error message
    - The user is not logged in

Test 3: Unsuccessful login due to account suspension

- **Preconditions:** The user's account is suspended.
- **Steps:**
    1. User enters the username and password.
    2. The system verifies the credentials against the registered user database.
- **Expected Result:**

- Error message
- The user is not logged in

Store Owner use cases:

Use case: **4.1** – **Inventory management** (Add product, Remove product, Update product)

1. **Actor**: Store Owner
2. **Pre-condition**: The store owner exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, the product is added/removed/updated according to the given parameters.
4. **Input parameters**: StoreID, Product information (name, price, ...)
5. **Actions**:
    1) Store Owner attempts to add/remove/update a product in a specific shop.
    2) Authentication – the system checks that the action was requested by an actual store owner of the needed store.
    3) Inventory availability – the system checks that the product is available to the requested action in the inventory. (exists if the action is remove/update, not exists if the action is add)
    4) Validity – the system checks that the given parameters are valid (positive,non-empty) and follow the standard rules of the store.
    5) The system updates the item in the inventory.
    6) The system returns a matching success response to the user.
6. **Alternatives/Extensions**:
    2) If the authentication failed, a matching error message will be returned and the action will fail.
    3) If the item is not available in the inventory, a matching error message will be returned and the action will fail.
    4) If the given parameters are invalid (for example zero or empty) or do not follow the standard rules of the store, a matching error message will be returned and the action will fail.

7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The requesting store owner does not have the right permissions (is not a store owner) in the requested store	Invaild StoreID, valid product information	Failed authentication, a matching error message will be returned.	Bad
The store owner is requesting to remove a non-existent product	Valid StoreID, valid product information	Failed inventory availability, a matching error message will be returned.	Bad
The store owner is requesting to add a new product with invalid product information (quantity -1)	Valid StoreID, invalid product information	Failed validity check, a matching error message will be returned.	Bad
The store owner is requesting to update an existing product, with valid product information	Valid StoreID, valid product information	Success, the product will be updated in the inventory.	Good

Use case: **4.2** – **Update store policies** (Update purchase policy, Update discount policy)

1. **Actor**: Store Owner
2. **Pre-condition**: The store owner exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, the store's purchase/discount policy is updated according to the given parameters.
4. **Input parameters**: StoreID, Purchase/Discount policy information.
5. **Actions**:
    1) Store Owner attempts to update the purchase/discount policy of a specific shop.
    2) Authentication – the system checks that the action was requested by an actual store owner of the needed store.
    3) Validity – the system checks that the given parameters are valid (positive,non-empty) and follow the standard rules of the store.
    4) The system updates the purchase/discount policy of the store.
    5) The system returns a matching success response to the user.
6. **Alternatives/Extensions**:
    2) If the authentication failed, a matching error message will be returned and the action will fail.
    3) If the given parameters are invalid (for example zero or empty) or do not follow the standard rules of the store, a matching error message will be returned and the action will fail.
7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The requesting store owner does not have the right permissions (is not a store owner) in the requested store	Invaild StoreID, valid discount policy information	Failed authentication, a matching error message will be returned.	Bad
The store owner is requesting to update with invalid discount policy information (quantity -1, discount -300%)	Valid StoreID, invalid discount policy information	Failed validity check, a matching error message will be returned.	Bad
The store owner is requesting to update the discount policy of a store, with valid discount policy information	Valid StoreID, valid discount policy information	Success, the discount policy of the store will be updated.	Good

Use case: **4.3** – **Store Owner Appointment** (Composite: Appointment Offer, Appointment Approval/Rejection)

Use case: **4.3.1** – **Store Owner Appointment Offer**

1. **Actor**: Store Owner, Subscriber
2. **Pre-condition**: The store owner exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, the selected subscriber will receive an appointment offer and will need to accept/reject the offer in the next part of the use case.
4. **Input parameters**: StoreID, username of the subscriber to be appointed as store owner.
5. **Actions**:
    1) Store Owner attempts to appoint a subscriber to be a store owner of a specific store.
    2) Authentication – the system checks that the action was requested by an actual store owner of the needed store.
    3) Validity – the system checks that the given parameters are valid – the given username corresponds to an actual subscriber of the market, the subscriber is not already a store owner of the given store, etc.
    4) The system sends an store owner appointment offer to the selected subscriber.
    5) The system returns a matching success response to the user.
6. **Alternatives/Extensions**:
    2) If the authentication failed, a matching error message will be returned and the action will fail.
    3) If the given parameters are invalid (for example if the given username doesn't belong to an actual subscriber, or if the subscriber is already a store owner of the store), a matching error message will be returned and the action will fail.

7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The requesting store owner does not have the right permissions (is not a store owner) in the requested store	Invaild StoreID, valid subscriber username	Failed authentication, a matching error message will be returned.	Bad
The store owner is requesting to appoint a store owner with invalid username (does not belong to a market subscriber)	Valid StoreID, invalid subscriber username	Failed validity check, a matching error message will be returned.	Bad
The store owner is requesting to appoint a store owner of a store, with valid subscriber username of a subscriber that isn't already a store owner of the specified store	Valid StoreID, valid subscriber username	Success, the subscriber will receive a store owner appointment offer, which will be accepted/rejected by him in the second part of the appointment use case.	Good

Use case: **4.3.2** – **Store Owner Appointment Approval/Rejection**

1. **Actor**: Subscriber, Store Owner
2. **Pre-condition**: The subscriber exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, if the subscriber accepted the appointment offer, the selected subscriber will become an owner of the selected store, marking the new owner as appointed by the previous owner.
4. **Input parameters**: The appointment offer id, approval/rejection of the selected subscriber to the given offer.
5. **Actions**:
    1) Subscriber is notified about an incoming appointment offer.
    2) Subscriber attempts to accept/reject the appointment offer.
    3) Authentication – the system checks that the action was requested by a subscriber of the market.
    4) Validity – the system checks that the given parameters are valid (appointment offer id actually exists in the system and belongs to the requesting subscriber).
    5) If the subscriber accepted the appointment offer, the subscriber will become an owner of the given store, marking the new owner as appointed by the previous owner and the requesting store owner will be notified accordingly. Otherwise, if the offer is rejected, the system dismisses it and notifies the requesting store owner accordingly.
    6) The system returns a matching success response to the user.
6. **Alternatives/Extensions**:
    3) If the authentication failed, a matching error message will be returned and the action will fail.
    4) If the given parameters are invalid (appointment offer id doesn't exist in the system or doesn't belong to the requesting subscriber), a matching error message will be returned and the action will fail.

7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The user sent an approval request which did not originate from an actual market subscriber	Invaild appointment offer id, offer approval	Failed authentication, a matching error message will be returned.	Bad
The subscriber sent an approval request for an appointment offer that didn't belong to him	Invalid appointment offer id, offer approval	Failed validity check, a matching error message will be returned.	Bad
The subscriber approved an appointment offer that belonged to him	Valid appointment offer id, offer approval	Success, the subscriber will become an owner of the given store, marking the new owner as appointed by the previous owner and the original requesting store owner will be notified accordingly	Good

Use case: **4.6** – **Store Manager Appointment** (Composite: Appointment Offer, Appointment Approval/Rejection)

Use case: **4.6.1** – **Store Manager Appointment Offer**

1. **Actor**: Store Owner, Subscriber
2. **Pre-condition**: The store owner exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, the selected subscriber will receive an appointment offer and will need to accept/reject the offer in the next part of the use case.
4. **Input parameters**: StoreID, username of the subscriber to be appointed as store manager.
5. **Actions**:
    1) Store Owner attempts to appoint a subscriber to be a store manager of a specific store.
    2) Authentication – the system checks that the action was requested by an actual store owner of the needed store.
    3) Validity – the system checks that the given parameters are valid – the given username corresponds to an actual subscriber of the market, the subscriber is not already a store owner/store manager of the given store, etc.
    4) The system sends an store manager appointment offer to the selected subscriber.
    5) The system returns a matching success response to the user.
6. **Alternatives/Extensions**:
    2) If the authentication failed, a matching error message will be returned and the action will fail.
    3) If the given parameters are invalid (for example if the given username doesn't belong to an actual subscriber, or if the subscriber is already a store owner/store manager of the store), a matching error message will be returned and the action will fail.

7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The requesting store owner does not have the right permissions (is not a store owner) in the requested store	Invaild StoreID, valid subscriber username	Failed authentication, a matching error message will be returned.	Bad
The store owner is requesting to appoint a store manager with invalid username (does not belong to a market subscriber)	Valid StoreID, invalid subscriber username	Failed validity check, a matching error message will be returned.	Bad
The store owner is requesting to appoint a store manager of a store, with valid subscriber username of a subscriber that isn't already a store owner or a store manager of the specified store	Valid StoreID, valid subscriber username	Success, the subscriber will receive a store manager appointment offer, which will be accepted/rejected by him in the second part of the appointment use case.	Good

Use case: **4.6.2** – **Store Manager Appointment Approval/Rejection**

1. **Actor**: Subscriber, Store Owner
2. **Pre-condition**: The subscriber exists in the system and is logged in to the system.
3. **Post-conditions**: In the case of success, if the subscriber accepted the appointment offer, the selected subscriber will become a manager of the selected store, marking the new manager as appointed by the previous owner.
4. **Input parameters**: The appointment offer id, approval/rejection of the selected subscriber to the given offer.
5. **Actions**:
    1) Subscriber is notified about an incoming appointment offer.
    2) Subscriber attempts to accept/reject the appointment offer.
    3) Authentication – the system checks that the action was requested by a subscriber of the market.
    4) Validity – the system checks that the given parameters are valid (appointment offer id actually exists in the system and belongs to the requesting subscriber).
    5) If the subscriber accepted the appointment offer, the subscriber will become a manager of the given store, marking the new manager as appointed by the requesting store owner, and the store owner will be notified accordingly. Otherwise, if the offer is rejected, the system dismisses it and notifies the requesting store owner accordingly.
    6) The system returns a matching success response to the user.
6. **Alternatives/Extensions**:
    5) If the authentication failed, a matching error message will be returned and the action will fail.
    6) If the given parameters are invalid (appointment offer id doesn't exist in the system or doesn't belong to the requesting subscriber), a matching error message will be returned and the action will fail.

7. **Acceptance tests – scenarios**:

Scenario Description	Parameters	Expected Result	Scenario
The user sent an approval request which did not originate from an actual market subscriber	Invaild appointment offer id, offer approval	Failed authentication, a matching error message will be returned.	Bad
The subscriber sent an approval request for an appointment offer that didn't belong to him	Invalid appointment offer id, offer approval	Failed validity check, a matching error message will be returned.	Bad
The subscriber approved an appointment offer that belonged to him	Valid appointment offer id, offer approval	Success, the subscriber will become a manager of the given store, marking the new manager as appointed by the requesting store owner, and the store owner will be notified accordingly	Good

Use case: **4.7** – **Store Manager Permission Management**

Actors:

Store owner

Preconditions:

The store owner exists and logged in and the store exists.

Postconditions:

1. The selected manager's permissions have been customized according to the settings defined by the store owner.
2. The manager can now access and perform only the tasks and functions permitted by their customized permissions.

Input parameters: StoreID, MangerID, removed permissions, added permissions.

Main Scenario:

1. **Authentication** – the system checks that the action was requested by an actual store owner of the needed store.
2. **Selecting a manager –** the store owner chooses a store manager that he has appointed.
3. The system checks that the chosen manager was appointed by the requesting store owner.
4. the store owner add/remove permissions from the selected store manager.
5. The system checks that the permissions are valid
6. The system updates the manager's permissions with the customized permissions.

Alternatives/Extensions:

1. If the authentication failed, a matching error message will be returned and the action will fail.
2. If the chosen store manager wanst appointed by the store owner, a matching error message will be returned and the action will fail
3. If the permissions are invalid (unrecognized to the system/ can't be assigned to store manager, a matching error message will be returned and the action will fail

Test 1: Successful Customization of Manager Permissions

Input: StoreID, ManagerID, removed permissions, added permissions.

Preconditions: The store owner is logged in and exists.

Steps:

1. Store owner selects a manager for permission customization.
2. Removed and added permissions are specified.
3. Permissions are valid and can be assigned to the manager.

Expected Outcome:

- The selected manager's permissions are updated according to the provided settings.
- The manager can access and perform only the tasks and functions permitted by their customized permissions.

Test 2: Authentication Failure

Input: StoreID, ManagerID, removed permissions, added permissions.

Preconditions: The store owner is not logged in or not exist.

Steps:

Store owner attempts to customize manager permissions.

Expected Outcome:

- Error message
- The action fails, and no changes are made to the manager's permissions.

Test 3: Invalid Manager Selection

Input: StoreID, ManagerID, removed permissions, added permissions.

Preconditions:

The store owner is logged in and is an owner of the store.

Steps:

Store owner selects a manager that was not appointed by them.

Expected Outcome:

- Error message
- The action fails, and no changes are made to the manager’s permissions.

Test 4: Invalid Permissions

Input: StoreID, ManagerID, removed permissions, added permissions.

Preconditions:

The store owner is logged in and is an owner of the store.

Steps:

Store owner attempts to assign invalid or unrecognized permissions to the manager.

Expected Outcome:

- Error message
- The action fails, and no changes are made to the manager’s permissions.

Test 5: Permissions Conflict

Input: StoreID, ManagerID, removed permissions, added permissions.

Preconditions:

The store owner is logged in and is an owner of the store.

Steps:

Store owner attempts to assign permissions that conflict with existing permissions of the manager (adding an existing permission for example).

Expected Outcome:

- Error message
- The action fails, and no changes are made to the manager’s permissions.

Use case: **4.9** – **Store Manager Permission Management**

Actors:

Store founder.

Preconditions:

The store founder exists and logged in and the store exists.

Postconditions:

1. The store is closed and becomes inactive.
2. Access to the closed store's information is restricted to store owners and system administrators.
3. Store owners and managers receive a notification about the closure, but their appointment remains unchanged.
4. Store's products will no longer apear while searching for products.

Input parameters: StoreID

Main Scenario:

1. **Authentication** – the system checks that the action was requested by the store founder of the needed store.
2. The system checks that the store is currently open
3. The system change the store status to inactive
4. The system sends a notification informing the store owners and managers of the store closure.

Alternatives/Extensions:

1. If the authentication failed, a matching error message will be returned and the action will fail.
2. If the store is allready closed, a matching error message will be returned and the action will fail.

Test 1: Successful Closure of the Store

Input: StoreID

Preconditions: The store founder is logged in and is the founder of the store.

Steps:

Store founder initiates the closure of the store.

Expected Outcome:

- The store status is changed to inactive.
- Access to the closed store's information is restricted to store owners and system administrators.
- Store owners and managers receive a notification about the closure, but their appointment remains unchanged.
- Store's products will no longer appear while searching for products.

Test 2: Authentication Failure

Input: StoreID

Preconditions:

The store founder is not logged in or is not the founder of the store.

Steps:

Unauthorized user attempts to close the store.

Expected Outcome:

- Error message
- The action fails, and the store status remains unchanged.

Test 3: Attempting to Close an Already Closed Store

Input: StoreID

Preconditions:

The store founder is logged in and is an owner of the store.

The store is already closed.

Steps:

1. Store founder closes the store.
2. Store founder attempts to close the store again.

Expected Outcome:

- Error message
- The action fails, and the store status remains unchanged.

Use case: **4.11** – **Request for Information on Store Roles**

Actors:

Store owner.

Preconditions:

The store owner exists and logged in and the store exists.

Postconditions:

The store owner has access to detailed information about the roles in their store.

Input parameters: StoreID

Main Scenario:

1. **Authentication** – the system checks that the action was requested by a store owner of the needed store.
2. The system retrieve inforamtion about all the roles in the store.

Alternatives/Extensions:

If the authentication failed, a matching error message will be returned and the action will fail.

Test 1: Successful Retrieval of Store Roles Information

Input: StoreID

Preconditions:

The store owner is logged in and is an owner of the store.

Steps:

Store owner initiates a request for information on store roles.

Expected Outcome:

The system retrieves detailed information about all the roles in the store.

Test 2: Authentication Failure

Input: StoreID

Preconditions:

The store owner is not logged in or is not an owner of the store.

Steps:

Unauthorized user attempts to request information on store roles.

Expected Outcome:

- Error message
- The action fails, and the requested information is not retrieved.

Use case: **4.11** – **Request for Information on manager permissions.**

Actors:

Store owner.

Preconditions:

The store owner exists and logged in and the store exists.

Postconditions:

The store owner has access to detailed information about the roles in their store.

Input parameters: StoreID, ManagerID

Main Scenario:

1. **Authentication** – the system checks that the action was requested by a store owner of the needed store.
2. The system checks that the manager exists and is a manager in the store
3. The system retrieve the manager permissions in the store.

Alternatives/Extensions:

1. If the authentication failed, a matching error message will be returned and the action will fail.
2. If the manager doesn't exists or is not a manager of the store, a matching error message will be returned and the action will fail.

Test 1: Successful Retrieval of Manager Permissions

Input: StoreID, ManagerID

Preconditions: The store owner is logged in and is an owner of the store. The manager exists and is assigned as a manager in the store.

Steps:

Store owner initiates a request for information on manager permissions.

Expected Outcome:

- The system retrieves information about the permissions assigned to the specified manager in the store.

Test 2: Authentication Failure

Input: StoreID, ManagerID

Preconditions:

The store owner is not logged in or not an owner of the store.

Steps:

Unauthorized user attempts to request information on manager permissions.

Expected Outcome:

- Error message
- The action fails, and the requested information is not retrieved.

Test 3: Manager Does Not Exist or Is Not a Manager in the Store

Input: StoreID, ManagerID

Preconditions: The store owner is logged in and is an owner of the store.

Steps:

1. Store owner initiates a request for information on manager permissions.
2. Store owner specifies a ManagerID that does not exist or is not assigned as a manager in the store.

Expected Outcome:

- Error message
- The action fails, and the requested information is not retrieved.

Use case: **4.13** – **Purchase History Retrieval.**

Actors:

Store owner.

Preconditions:

The store owner exists and logged in and the store exists.

Postconditions:

The store owner has access to detailed purchase history information.

Input parameters: StoreID

Main Scenario:

1. **Authentication** – the system checks that the action was requested by a store owner of the needed store.
2. The system retrieves the purchase history of the store.

Alternatives/Extensions:

1. If the authentication failed, a matching error message will be returned and the action will fail.
2. If there were no purchases from the store, a message telling there was no purchases will be returned.

Test 1: Successful Retrieval of Purchase History

Input: StoreID

Preconditions:

The store owner is logged in and is an owner of the store.

The store exists.

Steps:

Store owner initiates a request to retrieve purchase history.

Expected Outcome:

- The system retrieves purchase history for the specified store.

Test 2: Authentication Failure

Input: StoreID

Preconditions:

The store owner is not logged in or not an owner of the store.

Steps:

Unauthorized user attempts to retrieve purchase history.

Expected Outcome:

- Error message
- The action fails, and the requested information is not retrieved.

Test 3: No Purchase History Available

Input: StoreID

Preconditions:

The store owner is logged in and is an owner of the store.

The store exists

Steps:

Store owner initiates a request to retrieve purchase history.

Expected Outcome:

- Error message
- The action fails, and the requested information is not retrieved.

Use case: **5 –store manager permitted actions.**

Actors:

Store manager.

Preconditions:

The store manager exists and logged in and the store exists.

Postconditions:

The requested action is performed.

Input parameters: StoreID, action.

Main Scenario:

1. **Authentication** – the system checks that the action was requested by a store manager of the needed store.
2. The system checks if the action is within his permissions.
3. The system perform that action

Alternatives/Extensions:

3. If the authentication failed, a matching error message will be returned and the action will fail.
4. If the action is not within the store manager permissions, a matching error message will be returned and the action will fail.

Test 1: Successful Execution of Permitted Action

Input: StoreID, action

Preconditions:

The store manager is logged in and is a manager of the store.

The store exists.

Steps:

Store manager initiates the requested action.

Expected Outcome:

The system successfully performs the requested action

Test 2: Authentication Failure

Input: StoreID, action

Preconditions:

The store manager is not logged in or not a manager of the store.

Steps:

Unauthorized user attempts to initiate the requested action.

Expected Outcome:

- Error message
- The action fails, and the requested information is not retrieved.

Test 3: Action Not Within Store Manager Permissions

Input: StoreID, action

Preconditions:

The store manager is logged in and is a manager of the store.

The store exists.

The requested action is not within the store manager's permissions.

Steps:

Store manager attempts to initiate the requested action.

Expected Outcome:

- Error message
- The action fails, and the requested information is not retrieved.

Use Case: Change Purchase Policy

Actors:

- Store Owner

Preconditions:

- The store owner exists and is logged in.
- The store exists

Postconditions:

1. The purchase policy of the store is updated according to the store owner's configurations.

Input Parameters:

- User token
- StoreID
- Purchase policy

Main Scenario:

1. **Authentication**:
    - The system verifies that the action is being requested by a legitimate store owner.
2. **Select and Modify Policies**:
    - The store owner selects the types of purchase policy they wish to modify.
3. **Save Changes**:
    - The system changes the updated policies.

Extensions/Alternatives:

1. **Invalid Authentication**:
    - If the store owner is not authenticated an error message is displayed and the action is not completed.
2. **Validation Failure**:

- If the new policy settings is invalid, an error message is displayed, and the changes are not saved.

Test 1: Successful Policy Change:

- **Parameters**: Valid user token, Valid StoreID, Valid Purchase Types
- **Expected Result**: The purchase policy is updated successfully, and the changes are applied to the store's products.

Test 2: Invalid Authentication:

- **Parameters**: Invalid user token, valid StoreID, Valid Purchase Types, Valid Discount Policies
- **Expected Result**: Error message, The policies are not changed.

Test 3: Validation Failure:

- **Parameters**: Valid StoreID, Invalid Purchase policy
- **Expected Result**: Error message, The policies are not changed.

Use Case: Relinquish Ownership

Actors:

- Store Owner (Non-founder)

Preconditions:

- The store owner (non-founder) is logged in.
- The store exists.

Postconditions:

1. The store owner is no longer an owner of the store.
2. All appointments (owners and managers) made by this store owner are removed.

Input Parameters:

- StoreID
- OwnerID

Main Scenario:

1. **Authentication**:
    - The system verifies that the action is being requested by a legitimate store owner (non-founder).
2. **Initiate Relinquish Ownership**:
    - The store owner selects the option to relinquish ownership.
3. **Remove Appointments**:
    - The system removes all appointments (owners and managers) made by the store owner.
4. **Notification**:
    - The system notifies the store owner of the successful relinquishment and informs other affected parties about the changes in appointments.

Extensions/Alternatives:

1. **Invalid Authentication**:
    - If the store owner is not authenticated an error message is displayed, and the action is not completed.

Test 1: Successful Relinquishment of Ownership:

- **Parameters**: Valid StoreID, Valid OwnerID
- **Expected Result**: The ownership is relinquished successfully, and all appointments made by the store owner are removed. Notifications are sent to relevant parties.

Test 2: Invalid Authentication:

- **Parameters**: Invalid OwnerID, Valid StoreID
- **Expected Result**: Error message, The action is not completed.

Use Case: Suspend User

Actors:

- System Administrator

Preconditions:

- The system administrator is logged in
- The user to be suspended exists in the system.

Postconditions:

1. The selected user is suspended for a specified period or permanently.
2. The suspended user can only perform view operations in the system.

Input Parameters:

- AdminID
- UserID
- Suspension Duration (temporary or permanent)

Main Scenario:

1. **Authentication**:
    - The system verifies that the action is being requested by a legitimate system administrator.
2. **Initiate Suspension**:
    - The system administrator initiates the suspension process with user UserID.
3. **Specify Suspension Duration**:
    - The system administrator specify the duration of the suspension (temporary with a specified time period or permanent).
4. **Update User Status**:
    - The system updates the user's status to suspended

5. **Notification**:
    - The system notifies the user of their suspension and the terms of the suspension (duration and restrictions).

Extensions/Alternatives:

1. **Invalid Authentication**:
    - If the system administrator is not authenticated an error message is displayed, and the action is not completed.

Acceptance Tests:

1. **Successful User Suspension**:
    - **Parameters**: Valid AdminID, Valid UserID, Valid Suspension Duration
    - **Expected Result**: The user is suspended successfully, the user's status is updated, and the user is notified. The user can only perform view operations in the system.
2. **Invalid Authentication**:
    - **Parameters**: Invalid AdminID, Valid UserID, Any Suspension Duration
    - **Expected Result**: Error message, The action is not completed.

Use Case: Cancel User Suspension

Use Case ID: 7.4

Use Case Name: Cancel User Suspension

Actors:

- System Administrator

Preconditions:

- The system administrator is logged in
- The user to be unsuspended exists in the system

Postconditions:

1. The selected user's suspension is lifted.
2. The user regains access to all previously allowed system functionalities.

Input Parameters:

- AdminID
- UserID

Main Scenario:

1. **Authentication**:
    - The system verifies that the action is being requested by a legitimate system administrator.
2. **Initiate Unsuspension**:
    - The system administrator initiates the unsuspension process with the user UserID.
3. **Update User Status**:
    - The system updates the user’s status to active
4. **Notification**:
    - The system notifies the user that their suspension has been lifted and they now have full access to the system functionalities.

Extensions/Alternatives:

1. **Invalid Authentication**:
    - If the system administrator is not authenticated an error message is displayed, and the action is not completed.
2. **User Not Found**:
    - If the specified UserID is not currently suspended, an error message is displayed, and the action is not completed.

Acceptance Tests:

1. **Successful User Unsuspension**:
    - **Parameters**: Valid AdminID, Valid UserID
    - **Expected Result**: The user is successfully unsuspended, the user's status is updated to active, and the user is notified of the change.
2. **Invalid Authentication**:
    - **Parameters**: Invalid AdminID, Valid UserID
    - **Expected Result**: Error message, The action is not completed.
3. **User Not Found or Not Suspended**:
    - **Parameters**: Valid AdminID, UserID of a non-suspended user
    - **Expected Result**: Error message, The action is not completed.

Use Case: View User Suspensions

Actors:

- System Administrator

Preconditions:

- The system administrator is logged in

Postconditions:

1. The system administrator can view a list of suspended users along with details such as suspension end date.

Input Parameters:

- AdminID

Main Scenario:

1. **Authentication**:
    - The system verifies that the action is being requested by a legitimate system administrator.
2. **Request Suspension List**:
    - The system administrator requests to view the list of suspended users.
3. **Retrieve Suspension Data**:
    - The system retrieves and displays the suspension data for each suspended user, including:
        - End date of suspension

Extensions/Alternatives:

1. **Invalid Authentication**:
    - If the system administrator is not authenticated or does not have the necessary permissions, an error message is displayed, and the action is not completed.

Acceptance Tests:

1. **Successful View of Suspensions**:

- **Parameters**: Valid AdminID
- **Expected Result**: The system administrator successfully views a list of suspended users with details such as suspension end date.

2. **Invalid Authentication**:
    - **Parameters**: Invalid AdminID
    - **Expected Result**: Error message, The action is not completed.

Use Case 1: Implement Presentation Component Adaptation for Browsers

Use Case: Browser-Based Presentation Component

1. **Actors:** User (Visitor, Registered User, Admin)
2. **Pre-conditions:**
    1. User has accessed the market system via a web browser.
3. **Post-conditions:**
    - The user interface is displayed correctly on the browser.
    - User can navigate and use the market system features seamlessly.
4. **Input Parameters:**
    1. User token (if logged in)
    2. Browser type and version
    3. User actions (e.g., clicks, searches)
5. **Main Scenario:**
    1. User navigates to the market system website.
    2. The system detects the browser and loads the appropriate presentation layer.
    3. User interacts with the interface (e.g., searching for items, making purchases).
    4. The system ensures that the UI elements are responsive and function correctly.
6. **Alternatives:**
    1. If the browser is not supported, the system displays an error message suggesting alternative supported browsers.

Test 1: Successful Browser Adaptation

- **Parameters:**
    - Valid browser type and version
- **Expected Result:**
    - The user interface is displayed correctly and functions seamlessly.

Test 2: Unsupported Browser

- **Parameters:**
    - Unsupported browser type or outdated version
- **Expected Result:**
    - Error message suggesting alternative supported browsers.

Use Case 2: Password Privacy

Use Case: Mask Passwords in Input Fields

1. **Actors:** User
2. **Pre-conditions:**
    1. User is at a point where password input/display is required (e.g., login, account settings).
3. **Post-conditions:**
    - User's password is masked with '*' characters during input and display.
4. **Input Parameters:**
    1. User token (if logged in)
    2. Password input field
5. **Main Scenario:**
    1. User navigates to the login or account settings page.
    2. User enters their password in the provided field.
    3. The system masks the entered password with '*' characters.
    4. User submits the form.
    5. The system processes the login or settings update.
6. **Alternatives:**
    1. If the user opts to view the password, the system temporarily unmasks the password until the user disables this option.

Test 1: Successful Password Masking

- **Parameters:**
    - Valid password entry
- **Expected Result:**
    - Password is masked with '*' characters.

Test 2: Opting to View Password

- **Parameters:**
    - Valid password entry
    - User selects the option to view the password
- **Expected Result:**
    - Password is temporarily unmasked.

Use Case 3: User-Specific Interfaces

Use Case: Tailored User Interfaces Based on Roles

1. **Actors:** Visitor, Registered User, Admin
2. **Pre-conditions:**
    1. User is logged in with their respective role (if applicable).
3. **Post-conditions:**
    - User sees a tailored interface showing only the options and actions they are authorized to perform.
4. **Input Parameters:**
    1. User token
    2. User role (Visitor, Registered User, Admin)
5. **Main Scenario:**
    1. User logs into the market system.
    2. The system identifies the user role (Visitor, Registered User, Admin).
    3. The system displays the appropriate interface and available actions based on the user role.
    4. User performs allowed actions.
6. **Alternatives:**
    1. If a user attempts an unauthorized action, the system displays an error message.

Test 1: Correct Interface for Admin

- **Parameters:**
    - Valid admin token
- **Expected Result:**
    - Admin interface is displayed with all admin functionalities.

Test 2: Unauthorized Action Attempt by User

- **Parameters:**
    - Valid user token
    - Attempt to perform an admin action
- **Expected Result:**
    - Error message indicating unauthorized action.

Use Case 4: Explanation of Actions (Successes and Failures)

Use Case: Provide Explanations for Actions

1. **Actors:** User
2. **Pre-conditions:**
    1. User attempts an action that requires explanation (e.g., making a purchase).
3. **Post-conditions:**
    - User receives a clear explanation of the action's result.
4. **Input Parameters:**
    1. User token
    2. Action details (e.g., purchase, update)
5. **Main Scenario:**
    1. User initiates an action (e.g., purchasing an item).
    2. The system processes the action.
    3. If the action is successful, the system displays a success message.
    4. If the action fails, the system provides a detailed explanation of the failure (e.g., "Purchase failed due to insufficient funds").
6. **Alternatives:**
    1. System logs the reason for the failure for auditing purposes.

Test 1: Successful Action Explanation

- **Parameters:**
    - Valid action (e.g., valid purchase)
- **Expected Result:**
    - Success message indicating the action was completed successfully.

Test 2: Action Failure Explanation

- **Parameters:**
    - Invalid action (e.g., purchase with insufficient funds)
- **Expected Result:**
    - Error message explaining the reason for the failure.

Use Case 5: Defining and Editing Purchase and Discount Rules

Use Case: Define and Edit Store Rules for Discounts and Purchases

1. **Actors:** Store Owner
2. **Pre-conditions:**
    1. Store owner is logged into the system.
3. **Post-conditions:**
    - Discount and purchase rules are updated and active in the store.
4. **Input Parameters:**
    1. Store owner token
    2. Rule details (discount type, conditions, purchase rules)
5. **Main Scenario:**
    1. Store owner navigates to the discount and purchase rules section.
    2. Store owner defines or edits discount rules (e.g., "20% off on all dairy products").
    3. Store owner defines or edits purchase rules (e.g., "No alcohol sales after 11 PM").
    4. The system saves the changes and applies them to the store.
6. **Alternatives:**
    1. If the rule conflicts with existing policies, the system prompts the store owner to resolve the conflict.

Test 1: Successful Rule Definition

- **Parameters:**
    - Valid store owner token
    - Valid rule details
- **Expected Result:**
    - Rules are successfully defined and active in the store.

Test 2: Rule Conflict

- **Parameters:**
    - Valid store owner token
    - Conflicting rule details
- **Expected Result:**
    - Error message indicating the conflict and prompt to resolve it.

Use Case 6: Real-Time and Delayed Notifications

Use Case: Implement Notifications

1. **Actors:** User
2. **Pre-conditions:**
    1. User is logged into the system (for real-time notifications).
    2. System has a mechanism to track actions requiring notifications.
3. **Post-conditions:**
    - User receives notifications appropriately (in real-time or delayed).
4. **Input Parameters:**
    1. User token
    2. Notification trigger details
5. **Main Scenario:**
    1. User performs an action that triggers a notification (e.g., purchase).
    2. The system determines the type of notification (real-time or delayed).
    3. For real-time notifications, the system immediately displays the notification to the user.
    4. For delayed notifications, the system queues the notification and displays it when the user logs in next.
6. **Alternatives:**
    1. If the user has opted out of notifications, the system suppresses the notification.

Test 1: Real-Time Notification

- **Parameters:**
    - Valid user token
    - Action that triggers a real-time notification
- **Expected Result:**
    - Real-time notification is displayed immediately.

Test 2: Delayed Notification

- **Parameters:**
    - Valid user token
    - Action that triggers a delayed notification
- **Expected Result:**
    - Notification is queued and displayed upon next login.

Use Case: Persistent Data Storage

1. **Actors:** System
2. **Pre-conditions:**
    - None
3. **Post-conditions:**
    - System state is stored in an external database.
4. **Input Parameters:**
    1. ORM library configuration
    2. Database connection details
5. **Main Scenario:**
    1. The system performs an action that requires state saving.
    2. The ORM library manages data storage in the external database.
6. **Alternatives:**
    1. If the database connection fails, the system logs the error and retries.

Test 1: Successful Data Storage

- **Parameters:**
    - Valid ORM configuration
    - Valid database connection details
- **Expected Result:**
    - System state is successfully stored in the database.

Test 2: Database Connection Failure

- **Parameters:**
    - Invalid database connection details
- **Expected Result:**
    - Error is logged, and the system retries connection.

Use Case: System Resilience Against Failures

1. **Actors:** System
2. **Pre-conditions:**
    - System is running and interacting with external components.
3. **Post-conditions:**
    - System remains operational despite external failures.
4. **Input Parameters:**
    1. External system details
5. **Main Scenario:**
    1. System sends a request to an external component.
    2. External component fails to respond.
    3. System handles the failure and continues operation.
6. **Alternatives:**
    1. If an external system's behavior is unexpected, the system logs the error and switches to a backup process.

Test 1: Handling Communication Loss

- **Parameters:**
    - Valid system configuration
    - Simulated loss of communication with an external component
- **Expected Result:**
    - System logs the error and continues operating.

Test 2: Unexpected External System Behavior

- **Parameters:**
    - Valid system configuration
    - Simulated unexpected behavior from an external component (e.g., incompatible interface)
- **Expected Result:**
    - System logs the error and switches to a backup process.

Use Case: System Initialization from Configuration and Initial State Files

1. **Actors:** System
2. **Pre-conditions:**
    - Configuration file and initial state file are available.
3. **Post-conditions:**
    - System initializes correctly based on the provided files.
4. **Input Parameters:**
    1. Configuration file
    2. Initial state file
5. **Main Scenario:**
    1. System reads the configuration file.
    2. System connects to the specified database and external systems.
    3. System reads the initial state file and performs the described use cases.
6. **Alternatives:**
    1. If any action in the initial state file is illegal, the system logs an error and aborts initialization.

Test 1: Successful Initialization

- **Parameters:**
    - Valid configuration file
    - Valid initial state file
- **Expected Result:**
    - System initializes correctly and is in the required state.

Test 2: Initialization Failure Due to Illegal Action

- **Parameters:**
    - Valid configuration file
    - Initial state file with an illegal action
- **Expected Result:**
    - System logs an error and aborts initialization.