**Use Cases**

**System Use Cases:**

Initialize a Market:

Actor: Member

Pre-Condition: None

Parameters:

* member\_details
* market\_details

Actions:

1. System request for member \_details and market\_details.
2. creates a new market and set the member as System manager.
3. System presents all existing service options.
4. member chooses external services.
5. System shows the member whether the actions succeed or not.
6. If the action succeeds – the system shows the current Market details.

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Member | Member |
| Parameters | good visitor details and good market details | bad visitor details or/and bad market details |
| Result | the market is loaded | the system displays an error message and asks for new details |

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | External Service | External Service |
| Parameters | External Service is fully connected. | Failed to Connect External Service |
| Result | System Creates new Market | The system informs member, requests for new Service |

Change market connection to External Service:

Actor: Member

Pre-Condition:

* Member must be a System Manager

Parameters: new\_external\_service

Actions:

1. System manager chooses to replace specific external service.
2. System asks the system manager for the new external service.
3. User inserts new\_external\_service.
4. System tries to connect to new\_external\_service;
5. If succeed
   1. System change the service implementor to the new external service.
6. shows approving or failure messages to the user.

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | System manager | Market |
| Parameters | Good external service details | Bad external service details/ service which doesn’t exist |
| Result | The system is connected to the new external service | the system displays an error message that the service doesn’t exist |

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Member | System |
| Parameters | The Member is the System manager | The Member isn’t the System Manager |
| Result | The system starts the process | System Notifies of impossible action |

Payment:

Actor: Acquisition use case

Pre-Condition:

* shopping Cart must not be empty.

Finish Condition:

* result message has been received from the external service.

Parameters:

* payment\_details
* supply address
* shopping\_cart
* user\_details

Actions:

1. for each shopping basket in shopping\_cart:
   1. calculate basket use case
2. if any of the shopping basket calculation fails:
   1. notify the visitor of the error
   2. finish the process
3. sums all basket values
4. System checks if Supply Service can supply
5. If not -> system notifies the user he should try again later
6. System transfers request to the current payment service and checks if possible
7. Receives response res
8. Returns res

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Visitor | System |
| Parameters | Good payment and user details and good shopping cart | Illegal payment details |
| Result | Payment succeeds | the system displays an error message about the payment details |

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | External Service | External Service |
| Parameters | External Service gives a proper service | External Service cannot give proper service |
| Result | System is served | Sends an error to the System |

Calculate basket use case:

Actor: Payment use case

Pre-Condition:

Parameters:

* Shopping\_basket

Actions:

1. For every shop’s purchase policy:
   1. Check shopping\_basket is possible
   2. If not -> process fails.
2. For every product in Shopping\_basket:
   1. Gets shop’s product discount
   2. Chooses the best discount
3. For every shop’s general discount:
   1. Gets best discount according to shop’s policy
4. Calculates the overall value of the basket – including discounts
5. Returns the value

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Shopping\_basket | Shop |
| Parameters | Stands purchase policy | Exceed purchase policy |
| Result | The shop allows buying the products | Shop denies buying the products |

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | shop | System |
| Parameters | The shop has all products | The shop does not have all products by that time |
| Result | The shop allows buying the products | The process fails - sends a proper message to the visitor |

Supply:

Actor: Acquisition use case

Pre-Condition:

* Purchase of any kind has been completed

Parameters:

* shopping\_cart
* user\_details

Actions:

1. System transfers supply requests including shopping\_cart and user\_details to the current supply service.
2. Sends notifications to the user.

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | System | System |
| Parameters | Good shopping cart and user details | Shopping cart which doesn’t match the user details |
| Result | Supply passes to be the responsibility of the supply system | the system displays an error message about the unmatched shopping cart |

Shop manager’s live Notification for purchase:

Actor: purchase use case

Pre-Condition:

* Purchase made

Parameters:

* message

Actions:

1. for all listeners in shop’s purchase listeners:
   1. notify the listener with the message
   2. listener adds the message to its notifications box.

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | System | System |
| Parameters | Good message | Null |
| Result | The message is shown/saved | the system displays an error message about the message |

Offline Notification for account:

Actor: System

Pre-Condition:

* Existing member

Parameters:

* message

Actions:

1. notify Member with message
2. Member adds the notification to its notification box
3. Member is logging in
4. Member chooses to see notifications
5. System shows all notifications in its notification box, including the previous message

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | System | System |
| Parameters | Good message | Null |
| Result | The message is saved and displayed the hen user is logged in | the system displays an error message about the message |

**User Use Cases:**

Guest Log in:

Actor: user

Pre-Condition: None

Parameters: None

Actions:

1. User starts using the system
2. System creates new visitor
   1. System creates a new shopping cart
3. System presents all guest options

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | User | User |
| Parameters | ---- | ---- |
| Result | ---- | ---- |

User leaves the Market:

Actor: User

Pre-Condition:

* User is already a visitor in the market

Parameters: user\_id

Actions:

1. If the User is not logged in:
   1. User closes the application
   2. The system removes the user’s shopping cart.
   3. System removes temporary user (please look out for remove user use case).
2. Else (User is logged in)
   1. User chooses option “logout”
   2. System keeps the user’s state (shopping cart)
   3. User logs out
3. User is out of the system

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | User | User |
| Parameters | Logged in user\_id | Illegal user\_id |
| Result | The user is logged out or deleted | the system displays an error message about the user\_id |

Register:

Actor: User

Pre-Condition:

* User is already a visitor in the market
* User’s details are legal

Parameters: user details (name, password, etc.)

Actions:

1. User chooses to register
2. System requests user for details
3. If the user’s details are legal
   1. System creates a new user:
   2. System creates a new and empty shopping cart connected to the user
   3. System saves the user’s details
4. Else
   1. System presents an error

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | User | User |
| Parameters | Legal user details | A name that already exists in the system |
| Result | User-created successfully | the system displays an error message about the existing name |

Login:

Actor: User

Pre-Condition:

* User is already a visitor in the market

Parameters:

* Name
* Password
* Additional user’s logging options

Actions:

1. User chooses to log in
2. System asks for name, password, and user’s additional logging options
3. System validates each of the above
4. If all is correct:
   1. System imports the corresponding user
   2. System changes the user’s shopping cart
   3. User is now logged in
5. Else:
   1. System notifies the user of invalid answers.

Acceptance Tests Paths:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Good Scenario | Bad Scenario | Bad Scenario |
| Participants | User | User | User |
| Parameters | Legal user login details | The user that doesn’t exist | A password that unmatched the name |
| Result | The user logged in successfully | the system displays an error message about the unmatching password | the system displays an error message about the unmatching password |

Receive information about a shop:

Actor: User

Pre-Condition:

* Existing shop

Parameters:

* shop\_id

Actions:

1. User chooses a shop
2. System presents the shop’s option
3. User chooses “show info”
4. System presents shop’s info

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | User | User |
| Parameters | Legal shop identifier | A shop that doesn’t exist |
| Result | Shop information is displayed | the system displays an error message about the shop that doesn’t exist |

Product Search:

Actor: User

Pre-Condition:

* At least 1 Existing Shop
* Existing product

Parameters:

* Search\_text

Actions:

1. User chooses search option
2. System request what kind of search is required:
   1. “By category”
   2. “By name”
   3. “By tag”
3. User chooses option
4. System asks for seach\_text
5. System searches all products by option and search\_text
6. System shows all results

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | User | User |
| Parameters | Legal search\_text | Illegal search\_text (null/int…) |
| Result | Results of the search are displayed | the system displays an error message |

Save Products in Shopping Cart:

Actor: User

Pre-Condition:

* Product which is saved exists in the shop in the wanted amount

Parameters:

* Product\_id
* amount
* shop\_id

Actions:

1. User chooses “save”
2. User inserts the amount
3. If the inserted amount exists for this product:
   1. System adds the product to the user’s shopping basket of the current shop
4. Else:
   1. System presents an error

Acceptance Tests Paths:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Good Scenario | Bad Scenario | Bad Scenario |
| Participants | User | User | User |
| Parameters | Legal details | Negative amount | A product which doesn’t exist in the shop in the current amount |
| Result | The product is added to the shopping basket of the shop | the system displays an error message about the negative amount | the system displays an error message about the negative amount |

Show and change Shopping Cart:

Actor: User

Pre-Condition:

* User is already a visitor in the market
* Product to change is already in the shopping cart
* The relevant information if needed is legal

Parameters:

* Product\_to\_change
* Relevant information if needed

Actions:

1. User chooses to edit his\her shopping cart
2. System presents shopping cart
3. User chooses a product to edit
4. System shows product’s edit options (change amount or delete)
5. User chooses an option and edits
6. User inserts the relevant information if needed
7. If everything is legal
   1. Changes are saved and the shopping cart is updated
8. Else
   1. System presents an error

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | user | user |
| Parameters | Details of a valid product. | Details of a product that does not exist. |
| Result | The product is changed. | the system displays an error message and asks for new details. |

Acquisition:

Actor: User

Pre-Condition:

* User is already a visitor in the market

Parameters:

* shopping\_cart

Actions:

1. Visitor chooses to buy his/her shopping\_cart
2. System checks the feasibility of the payment (please look out for System’s Payment use case).
3. If is feasible:
   1. System transfers the shopping\_cart to the supply service (please look out for Supply use case) with user\_details
   2. The system waits for supply service response
   3. If the response is positive
      1. The system applies for payment with the payment service
      2. The system updates all shops which products in the cart bought from them about the products and the amount
      3. System cleans the shopping\_cart.
      4. System sends a notification to the buyer.
   4. Else
      1. System presents an error message
4. If is not feasible
   1. The system presents an error message

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | user | user |
| Parameters | Valid shopping cart | An empty shopping cart |
| Result | the Acquisition has been done. | the system displays an error message and asks for new details. |

Add a Bid:

Actor: User

Pre-Condition:

* User is already a visitor in the market
* The relevant item exists in the relevant shop
* The price offered is positive

Post-Condition:

* Bid sent to all shop owners to approve
* Bid is added to shop’s bids
* Bis is added to the user’s shopping cart

Parameters:

* Shop name
* Item id
* Price
* amount

Actions:

1. Visitor chooses to add a Bid on a specific item
2. Visitor chooses the price and the amount
3. System checks the parameters and if the bid is feasible
4. If is feasible:
   1. System creates a new Bid
   2. System sends notification to all relevant employees
   3. The bid is added to the visitor’s shopping cart
   4. System waits for employees to approve/reject/counter-bid
   5. If all the employees approved the bid:
      1. System sends notification to the visitor who added the bid.
      2. The bid is approved and when the visitor will buy the shopping cart the item which is in the bid will be added to the purchase.
      3. The shop will update its inventory.
   6. If one of the employees reject the bid:
      1. System sends notification to the visitor and the employees
      2. The bid is deleted from the shop and from the visitor’s shopping cart
   7. If one of the employees adds a counter bid:
      1. System sends notification to all employees to approve the counter bid
      2. All approvals are canceled
      3. System waits to all the employees to approve.
      4. If all the employees approved the counter bid:
         1. System sends notification to the visitor to approve the counter bid.
         2. If the visitor approves the counter bid – same as e
         3. If the visitor reject the counter bid – same as f
      5. If one of the employees reject the counter bid – same as f
      6. If one of the employees adds counter bid – starts g from the beginning
5. If is not feasible
   1. The system presents an error message

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | user | user |
| Parameters | Valid price, amount, shop and item | Invalid info |
| Result | the Bid is added. | the system displays an error message and asks for new details. |

**Shop Owner Cases:**

Stock Management:

Actor: User

Pre-Condition:

* User must be logged in.
* User must be the shop owner.
* If editing or removing an item – item exists in the shop
* If adding a new item – item doesn’t exist in the shop

Parameters: item’s new details

Actions:

1. User Chooses to edit products
2. User Chooses between options:
   1. Add a new item
   2. Remove an existing item
   3. Edit an existing item
3. If the user chose option b or c:
   1. system shows all existing items
   2. The user chooses an item
   3. The system shows the item’s details
   4. If the user chose option b (remove an existing item):
      1. System presents a way to delete the item
      2. System asks the user to approve the deleting
   5. Else (option c – edit an existing item):
      1. User chooses details to edit (price for example)
      2. The user inserts the updated details (item’s price for example)
      3. System asks the user to approve the change
      4. System changes the item’s detail accordingly
4. Else (user chose option a – add a new product):
   1. System asks the user for product details
   2. System checks legality of the item’s details
   3. If the item’s details are legal:
      1. System creates a new item and adds it to the shop items list
   4. Else:
      1. System presents an error

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Shop owner | Shop owner |
| Parameters | Valid product price. | Negative product price. |
| Result | The product has been updated. | change is not done. The system displays an error message and asks for new details. |

Add policy to shop:

Actor: User

Pre-Condition:

* User must be logged in
* User must be the shop owner or founder

Parameters: policy’s relevant new details

Actions:

1. User Chooses to add the policy
2. System presents the policy’s options
3. User chooses the relevant option
4. User inserts the relevant new policy details
5. If the updated policy conflicts with a consistent policy of the shop founder
   1. System presents an error message
6. Else
   1. System adds the policy accordingly

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Shop owner | Shop owner |
| Parameters | Valid details | Invalid details |
| Result | The policy added to shop. | The policy is not added to shop. An error message is displayed. |

Change Shop’s Policy:

Actor: User

Pre-Condition:

* User must be logged in
* User must be the shop owner or founder

Parameters: policy’s relevant new details

Actions:

1.User Chooses to edit the policy

2.System presents the editing options

3.User chooses the relevant option

4.User inserts the relevant new details

5.If the updated policy conflicts with a consistent policy of the shop founder

a. System presents an error message

6.Else

a. System updates the policy accordingly

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Shop owner | Shop owner |
| Parameters | Valid details | Invalid details |
| Result | The shop’s policy is changed. | The shop’s policy is not changed. An error message is displayed. |

Remove policy from shop:

Actor: User

Pre-Condition:

* User must be logged in
* User must be the shop owner or founder

Parameters: Policy to remove

Actions:

1.User Chooses to remove the policy

2.System presents the policy’s options

3.User chooses the relevant option

4.User chooses policy to remove

5.If all details legal

a. The system removes the policy

6. Else

a. System presents an error message

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Shop owner | Shop owner |
| Parameters | Valid details | Invalid details |
| Result | The policy removed from the shop. | The policy is not removed from the shop. An error message is displayed. |

Shop Owner Appointment:

Actor: User

Pre-Condition:

* User must be logged in
* User must be the shop owner
* User2 is an existing user
* User2 is not an owner of this shop yet

Parameters: User2

Actions:

1. User chooses to add a new shop owner
2. System requests for user2 info
3. User inserts user2 info
4. If the info is legal and the user is not yet the owner of the shop:
   1. System moves the request to all shop owners for overall consent
   2. The system creates a new Shop Owner (state).
   3. System adds user2 to user's appointments list.
5. If user2 is in the shop’s employees list:
   1. Changes user2 role (State) to shop owner
6. Else:
   1. Adds user2 to shop employees list with shop owner as role.

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Shop owners | Shop owners |
| Parameters | A user who is a member | A user that is not a member |
| Result | User2’s role is changed. | User2’s role is not changed. The system displays an error message. |

Shop Manager Appointment:

Actor: Shop Owner

Pre-Condition:

* Existing shop.
* User must be logged in
* User must be the shop owner
* User2 - Another Existing user
* User2 – not a shop manager or owner yet

Parameters: user2

Actions:

1. User chooses to add a new shop manager
2. System requests for user info
3. User inserts user2 info
4. If user2 info is legal and user2 is not a manager or an owner of this shop yet
   1. System moves the request to all shop owners for overall consent.
   2. If there is a consent:
      1. System creates a new shop manager(State).
      2. System adds user2 to user's appointments list
      3. System adds user2 to shop employees list with shop manager as a role
   3. Else
      1. System updates the user that there was no consent
5. Else:
   1. System presents an error

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Shop owners | Shop owners |
| Parameters | A user that is a member. | A user that is not a member. |
| Result | The shop owner becomes a shop manager successfully. | The shop owner stays a shop owner. The system displays an error message. |

Edit Manager Permission:

Actor: User

Pre-Condition:

* User must be logged in.
* User must be the shop owner
* User2 must be the shop manager
* User 2 is necessarily appointed by the user

Parameters:

* Permission to change(add or remove)
* User2 info

Actions:

1. User chooses to edit a shop manager’s permission
2. System request for the user's details
3. User inserts user2 info.
4. System shows all permissions with status
5. User chooses permission to edit
6. System adds or removes the permission in user2’s permission list.

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Shop owner | Shop owner |
| Parameters | Shop manager, legal permission | A user that is not a shop manager, permission |
| Result | The permission is updated. | The permission is not updated. The system displays an error message. |

Close a Shop:

Actor: User

Pre-Condition:

* User must be logged in
* User must be the shop founder

Parameters: shop\_id

Actions:

1. User chooses to close the shop
2. System changes the shop to be unavailable
3. For all managers and owners of shop:
   1. notifies the shop is closed
4. For all acquisitions by time:
   1. notifies the user the shop is closed
5. System saves the shop in history

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Shop founder | Shop founder |
| Parameters | Valid shop id | Shop id that does not exist |
| Result | Shop is closed | The system displays an error message. |

Show Employees Info:

Actor: User

Pre-Condition:

* User is logged in
* User is the shop owner

Parameters: None

Actions:

1. User chooses to show all employees’ info
2. for all employees in the shop employees list:
   1. Show user details.
   2. If manager – show manager’s permission.

Acceptance Tests Paths:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Good Scenario | Bad Scenario | Bad Scenario |
| Participants | Shop owner | Shop owner | Shop owner |
| Parameters | Valid shop id | Invalid shop id | The user is not the shop’s owner |
| Result | Details are displayed | Details are not displayed. Error message is displayed | Error message is displayed |

Show Acquisition History:

Actor: User

Pre-Condition:

* User must be logged in
* User must have permission to shop history

Parameters: none

Actions:

1. User chooses to show acquisition history
2. System imports shop history.
3. System presents the history

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | User | - |
| Parameters | - | - |
| Result | Show Acquisition History is displayed. | - |

**Shop Manager Cases:**

Read information and response:

Actor: User

Pre-Condition:

* User must be logged in
* User must be the shop’s manager
* User must have the permission

Parameters: message-id

Actions:

1. User chooses to show all information
2. User chooses a message
3. System shows the option
4. If the user chooses to respond
   1. User inserts his answer
   2. System removes the message from shop’s messages
   3. System notifies the relevant user with the response
5. Else
   1. The system stays with no changes

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | Shop managers | Shop managers |
| Parameters | Valid message-id | Invalid message-id |
| Result | The message is taken care of and removed. | The message is not removed. An error message is displayed. |

Respond a bid:

Actor: User

Pre-Condition:

* Bid added to shop
* User is the shop manager
* Manager has Approve a bid permission

Parameters: shop manager name, shop name, visitor name, item id

Actions:

1. User chooses to show all bids
2. User chooses a bid
3. System shows the option
4. If the user chooses to approve
   1. Shop manager status changes to true.
   2. If this is the last shop manger to approve – see add a bid info
   3. Else – The system waits to all the employee’s response
5. If the user chooses to reject - see add a bid info
6. If the user chooses to add a counter bid:
   1. System checks the counter bid is feasible
   2. If is feasible - see add a bid info
   3. Else - System presents an error message

Acceptance Tests Paths:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Good Scenario | Bad Scenario | Bad Scenario | Bad Scenario |
| Participants | Shop managers | Shop managers | Shop managers | Shop managers |
| Parameters | Valid parameters | Bid is not for the manager to approve | Shop manager does not have the right permission | User is not the shop manager |
| Result | As the above | No change. An error message is displayed. | No change. An error message is displayed. | No change. An error message is displayed. |

**System Manager Cases:**

Show Acquisition Info:

Actor: User

Pre-Condition:

* User must be logged in
* User must be a system manager

Parameters: shop\_id

Actions:

1. User chooses to get acquisition history by the shop
2. System request for shop details
3. Systems manager inserts shop\_id
4. System search for acquisitions in the shop acquisition history
5. System presents acquisition

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | System manager | System manager |
| Parameters | Valid shop id | Invalid shop id |
| Result | The details are displayed | The details are not displayed. An error message is displayed. |

**System notification service:**

Show Acquisition Info:

Actor: Market

Pre-Condition:

* An action has occurred that requires an update from some user.
* The user is logged in to the system, or the user is a member of the market.

Parameters: The message needed to be sent, the user connection.

Actions:

1. An action occurred in the system.
2. Systems need to update the user for the action.
3. System check if user is connected at the moment.
4. If he is, it sends the message in real time to the user.
5. If the user is not logged in and he is a market's member, the system will keep the message until the next time he will log in as member.
6. If the user is not logged in and not a system member, the system will ignore the message.

Acceptance Tests Paths:

|  |  |  |
| --- | --- | --- |
|  | Good Scenario | Bad Scenario |
| Participants | user | user |
| Parameters | User logged in | User is not logged in but is a member |
| Result | Message is sent to the user and shown in the UI | The system will keep the message until he will log in to the system as member. |