**Use Cases**

**System Use Cases:**

Initialize a Market:

Actor: user

Pre-Condition: None

Parameters:

* user\_details.
* market\_details.

Actions:

1. System request for user\_details and market\_details.
2. creates a new market and set the user as Market manager.
3. System presents all existing service options.
4. User choose supply\_external \_service, payment\_external\_service.
5. System shows the user whether the actions succeed or not.
6. System If action succeed - shows the current Market details.

Change market connection to External Service:

Actor: Market-Manager

Pre-Condition:

* user must be logged in.
* market must be initialized.
* all uses of the service must be finished.
* The new External Service must implement the corresponding Service Interface.

Parameters: new\_external\_service

Actions:

1. System checks for existing ongoing use of the current service.
2. if found – waits for all existing uses of the service to be done.
3. System ask the user for new the service
4. User inserts new\_external\_service.
5. System change the service implementor to new\_external\_service.
6. shows approving massage to user.

Payment:

Actor: user/ time

Pre-Condition:

* shoppingCart must not be empty.
* `` of any kind has been done

Finish Condition:

* result massage has been received from the external service.

Parameters:

* payment\_details
* shopping\_cart
* user\_details`

Actions:

1. System transfer request to the current payment service
2. Waits for service to finish and receives response res
3. If res is positive:
   1. transfer the shopping\_cart the supply service (please lookout for Supply use case) with user\_details.
   2. cleans the shopping\_cart.
   3. Sends notification to buyer.
4. If res is negative, tell shows the user the error message.

Supply:

Actor: system

Pre-Condition:

* Acquisition of any kind has been completed

Parameters:

* shopping\_cart
* user\_details

Actions:

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

Shop manager live Notification for acquisition:

Actor: System

Pre-Condition:

* Acquisition made.
* Existing shop

Parameters:

* message

Actions:

1. for all listener in shop’s acquisition listeners:
   1. notify listener with massage.
   2. if listener is logged in:
      1. pops (prints) new message.
   3. else:
      1. listener adds message to its notifications box.

Offline Notification for account:

Actor: System

Pre-Condition:

* notification has been triggered.
* existing user.
* user is logged off.

Parameters:

* message

Actions:

1. notify user with message
2. user adds notification to its notification box.
3. User logging in.
4. System shows all notification its notification box, including previous message.

**User Use Cases:**

Guest Log in:

Actor: user

Pre-Condition:

* Existing Market.

Parameters: none

Actions:

1. User starts using system
2. System creates new temporary user (Guest):
   1. System creates new shopping cart
3. System shows all guest options

User leaves the Market:

Actor: user

Pre-Condition:

* Existing Market.

Parameters: none

Actions:

1. User chooses option “logout” or just closes the application.
2. If User is guest:
   1. System removes the user’s shopping cart.
   2. System removes temporary user (please lookout for remove user use case).
3. User logs out

Register:

Actor: user

Pre-Condition:

* Existing Market.
* User logged in as guest

Parameters: user details

Actions:

1. User chooses to register
2. System request user for user details
3. System creates new user:
   1. Creates new and empty shopping cart.

Login:

Actor: user

Pre-Condition:

* Existing Market.
* Existing user

Parameters:

* Name
* Password
* Additional user’s logging options

Actions:

1. User to login.
2. System for name, password and user’s additional logging options
3. System validates each of the above.
4. If all correct:
   1. System imports the corresponding user.
   2. Changes the user’s shopping cart
5. Else:
   1. System notifies the user for invalid answers.

Receive information of a shop:

Actor: user

Pre-Condition:

* Existing shop

Parameters:

* *shop*

Actions:

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

Product Search:

Actor: user

Pre-Condition:

* At least 1 Existing Shop
* Existing product

Parameters:

* *Search\_text*

Actions:

1. user choose search option
2. System request what kind of search required:
   1. “by category”
   2. “by name”
   3. “by tag”
3. User chooses *option*
4. System ask for *seach\_text*
5. System search all product by *option* and *search text*.
6. System shows all results

Save Products in Shopping Cart:

Actor: user

Pre-Condition:

* Existing user (can be temporary)
* Existing Product
* Existing Shop

Parameters:

* Product
* amount
* shop

Actions:

1. user choose “save”.
2. user inserts amount.
3. If user does not have product list for the shop:
   1. System initializes new list for shop
4. System adds the product with its amount to user shopping cart.

Show and change Shopping Cart:

Actor: user

Pre-Condition:

* Existing user (can be temporary)

Parameters:

* *product*

Actions:

1. User choose to edit his\her shopping cart.
2. System prints shopping cart.
3. User choose *product* to edit.
4. System show *product* edit options.
5. User choose an option and edits.

Acquisition:

Actor: user

Pre-Condition:

* Existing user (can be temporary)
* Not empty shopping cart
* Existing shops.
* Existing products

Parameters:

* *shopping\_cart*

Actions:

1. If the acquisition isn’t immediate and user isn’t logged in (in case of acquisition by time of offline guest):
   1. Notify shops in *shopping\_cart* for missing user.
2. user choose to buy *shopping\_cart*
3. System applies payment (please lookout for System’s Payment use case).
4. Supply service returns *supply\_details*
5. System notifies all shops in *shopping\_car* of its bought products.
6. System cleans user’s shopping cart.
7. If user is logged in:
   1. System print to user the *supply\_details*
8. Else:
   1. System notify user with *supply\_details (please lookout for user’s Notification’s use case).*

**Shop Owner Cases:**

Stock Management:

Actor: Shop Owner

Pre-Condition:

* Existing shop.
* User must be logged in.
* User must be the shop owner.

Parameters: *product\_price.*

Actions:

1. User Choose edit products.
2. System present options:
   1. “add new product”
   2. “remove existing product”
   3. “edit existing product”
3. System shows all existing products.
4. User chooses a product.
5. System shows product’s details
6. User choose details to edit (price)
7. user inserts *product\_price*
8. System changes product’s price

Change Shop’s Policy:

Actor: Shop Owner

Pre-Condition:

* Existing shop.
* User must be logged in.
* User must be the shop owner.

Parameters: *discount\_policy.*

Actions:

1. User Choose edit policy.
2. System present options
3. User chooses to add new discount
4. System presents discount policy options
5. User chooses *option*
6. User inserts *discount\_policy.*
7. System adds policy to shop’s discount list.

Shop Owner 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 owner yet

Parameters: *user2*

Actions:

1. User choose to add new shop owner
2. System requests for user info.
3. User inserts user2
4. System creates new Shop Owner (state)
5. If user2 is in 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.

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 choose to add new shop manager.
2. System requests for user info.
3. User inserts *user2.*
4. System creates new shop\_manager.
5. Systems sets shop\_manager permissions to have:
   1. Receive information and response.
   2. Acquisition History
6. If *user2* is in shop’s employees list:
   1. Changes *user2* role (State) to shop manager.
7. Else:
   1. Adds *user2* to shop employees list with shop manger as role.

Edit Manager Permission:

Actor: shop owner

Pre-Condition:

* Existing shop.
* User must be logged in.
* User must be the shop owner.
* *User2* – existing show manager

Parameters:

* *Permission*
* *User2*

Actions:

1. User choose to edit manger permission
2. System request for user details
3. User inserts *user2*
4. System present edit options
5. User choose to delete permission
6. System shows all existing premissions
7. User choose *permission* to remove.
8. System removes permission from *user2’s* shop manager role’s permission list

Close a Shop:

Actor: shop founder

Pre-Condition:

* shop founder must be logged in
* shop must exist

Parameters: None

Actions:

1. shop founder choose to close the shop
2. System changes the shop to be unavailable
3. For all manager and owner in shop:
   1. notifies the shop closed
4. For all acquisition by time:
   1. notifies user the shop closed

Show Employees Info:

Actor: shop - owner

Pre-Condition:

* shop owner is logged in
* shop exists

Parameters: none

Actions:

1. show owner choose to show all employees info
2. for all employee in shop employee list:
   1. show user details.
   2. show user permission.

Show Acquisition History:

Actor: user

Pre-Condition:

* user must be logged in
* user must have permission to shop history
* shop exists

Parameters: none

Actions:

1. user chooses to show acquisition history
2. System imports shop history.
3. System shows history

**Shop Manager Cases:**

Read information and response:

Actor: shop managers

Pre-Condition:

* The shop manager must be logged in
* Shop must exist
* Shop must be available
* Shop manager must have the permission

Parameters: none

Actions:

1. User chooses to show all information
2. User chooses to a message
3. System shows option
4. User chooses to respond
5. User insertד his answer
6. System removes the message from shop’s messages
7. System notifies the user with the response

**System Manager Cases:**

Show Acquisition Info:

Actor: System-Manager

Pre-Condition:

* System manager must be logged in

Parameters: *shop*

Actions:

1. User choose to show acquisition history.
2. System presents history option.
3. User chooses acquisition history by shop.
4. System request for shop details.
5. User inserts *shop.*
6. System search acquisitions - shop acquisition history
7. System shows acquisitions

Initialize a Market:

Actor: Market-Manager

Pre-Condition: None

Finish Condition:

Parameters:

Actions: